(12) (R)

FINAL REPORT FOR CONTRACT N00014-89-J-1658

Project Director:	Donna Crystal Llewellyn
Initiation Date:	February 2, 1989
Termination Date:	September 30, 1992
Project Title:	Location and Distribution of Local Optima

11pp

PAPERS

Two papers appeared in refereed journals since the proposal was submitted:

"Finding Saddlepoints of Two-Person, Zero Sum Games," with Craig Tovey and Michael Trick. American Mathematical Monthly Volume 95, Number 10, December 1988. Pages 912-918.

"Local Optimization on Graphs," with Craig Tovey and Michael Trick. DIscrete Applied Mathematics Volume 23, 1989. Pages 157-178.

Two papers have been accepted for publication in referreed publications:

"Dividing and Conquering the Square," with Craig Tovey. To appear in Discrete Applied Mathematics.

"A Primal Dual Integer Programming Algorithm," with Jennifer Ryan. To appear in Discrete Applied Mathematics.

Two papers have been submitted for publication in referreed publications:

"The Bernoulli Salesman Problem: Asymptotic Analysis," with Linda Whitaker. Submitted to Mathematics of Operations Research.

"2-Lattice Polyhedra: Duality," with Shiow-yun Chang and John Vande Vate. Submitted to Journal of Combinatorial Theory.

Three papers are in the process of being written:

"The Bernoulli Salesman Problem: Nonasymptotic Analysis," with Linda Whitaker.

"Matching 2-Lattice Polyhedra: Extreme Points," with Shiow-yun Chang and John Vande Vate.

"One-Machine Generalized Precedence Constraint Scheduling Problems," with Erick Wikum and George Nemhauser.

STUDENTS

Two students have been supported on this grant:

Linda Whitaker: Graduated with PhD in August 1992; thesis title: "The Bernoulli Salesman;" current position: Visiting Assistant Professor at SUNY at Stony Brook.

Shiow-yun Chang: Expected graduation date of June 1993; expected thesis title: "2-Lattice Polyhedra."

OTHER

The project director has edited a special issue of Discrete Applied Mathematics on the subject of Local Optimization. The issue is currently at the publisher.

The Bernoulli Salesman Problem: Asymptotic Analysis

L.M. Whitaker
Harriman School for Management and Policy, SUNY at Stony Brook, Stony Brook, NY
and
D.C. Llewellyn
School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA

Abstract

In this paper, we present a probabilistic analysis of the time versus solution quality tradeoff of different implementations of a basic sequential edge exchange procedure for a Traveling Salesman Problem. Our TSP will be on a complete graph with edge weights assigned independently and identically according to a Bernoulli distribution. One implementation of the procedure is a generalization of the Lin-Kernighan heuristic. For this implementation, we find asymptotic performance guarantees which decrease geometrically as the depth of the search increases. The basic search procedure may be implemented using a 2-change neighborhood structure. This enables us to find an asymptotic performance guarantee for a 2-opt procedure.

Key Words: Traveling Salesman, Local Improvement, Random Graphs.

1 Introduction

We will be investigating a random TSP which we designate as the *Bernoulli Salesman* problem. This will be defined as the problem of finding a tour of smallest value on a complete graph, whose edge weights are assigned identically and independently. The weight of an edge follows a Bernoulli distribution, with the probability an edge has weight 0 being $p_0(n)$. We require that $p_0(n) = c/n$, $c^* \leq c \leq 1.1 \log n$, where c^* is a large constant. (At $c = 1.1 \log n$, a tour of length 0 almost always exists, which could be found by our methods, as well as those of other authors [2],[4].)

We investigate how well local optimization can perform on this problem. We will use an exchange method which preserves tour feasibility after every pair of exchanges. Different implementations of this procedure yield a 2-opt procedure and the Lin-Kernighan heuristic . A 2-opt local search algorithm uses a 2-change neighborhood where two tours are neighbors if they share all edges of a tour except two. Exchanging two edges in a tour for two new edges leads to a neighboring tour. The Lin-Kernighan algorithm (denoted here as the LK algorithm) does a series of edge exchanges; it is a variable k-change heuristic designed to avoid local optima which would trap other procedures like the 2-opt.

A related procedure is the algorithm HAM, used in Frieze [4], [2] to find Hamiltonian Cycles. This algorithm looks for Hamiltonian Cycles on a random graph. It is based upon the extension and rotation idea usually credited to Posa [11]. The algorithm finds a Hamiltonian

Cycle on this random graph with probability which approaches the probability the cycle exists. We use techniques based partly on this analysis to find asymptotic guaranteed tour values for different implementations of our edge exchange procedure.

Very few theoretical results have been found for local search procedures. Kern [7] has a nice result for the Euclidean based TSP, based on a uniform distribution for the city locations. He showed that with high probability, 2-opting will find a local optimum in polynomial time. Unfortunately, this result does not address the quality of the local optimum found.

Most TSP algorithms that yield to probabilistic analysis have the same underlying theme. The problem is split up into small subproblems which are solved to optimality by some brute force method (like dynammic programming). Then, these small subtours or paths are hooked together in some way which does not cause the tour value to increase at an explosive rate. Examples of these algorithms are Karp's disection algorithm [6], the patching algorithm for the asymmetric TSP [8], Frieze's TSP algorithm [4], and Steele's directed TSP algorithm [12]. All results of this type are asymptotic, they only hold when $n \rightarrow \infty$. The asymptotic results of these algorithms in some cases give tours that converge to the optimal tour value. However we have very little information that suggests these algorithms perform well in practice.

The LK procedure has good empirical backing. HAM performs well in the asymptotic sense. We tie these two sets of information together. In this paper, we present an asymptotic analysis of a procedure which generalizes 2-opting and the LK algorithm. Specifically, we find an asymptotic guarantee of solution quality as a function of the time requirement. This guarantee shows a geometric improvement with respect to the amount of time we are willing to spend. (See Theorem 5.2 and Corollary 5.1.) *We also present an asymptotic guarantee* for a 2-opt procedure

In our next paper, we complete the analysis by presenting empirical and nonasymptotic results for our local search procedure. Since local search procedures are so widely used, these results could be useful in practice, as well as theoretically interesting and pleasing.

The organization of this paper is as follows. In Section 2 we introduce the important notational assumptions of our work. In Section 3 we describe our algorithm in detail. We analyze the algorithm in Section 5. In Section 6 we show how our algorithm can be modified to be a strict 2-opt procedure, and how to handle aymmetric instances. Several proofs are deferred to the appendix.

2 Notation

For ease of presentation the basic notation used will be given here.

$\mathcal{G}(n, c/n)$	The set of all complete graphs on vertices $V = \{1, 2, .., n\}$ with edge weights assigned independently and identically, and P(edge e has weight 0) $= p_0(n) = c/n$.
$G_{n,c/n}$	A random graph in $\mathcal{G}(n, c/n)$.
c	$c^* < c \leq 1.1 \log n,$; c^* is a large constant ($c(= c(n))$ may be a funtion of n)
E	The set of edges of G, where $G = (V, E)$ is distributed like $G_{n,c/n}$.
E_0	$\{e \in E$: edge e has weight 0$\}$.
0-edge	An edge of weight 0
1-edge	An edge of weight 1.
$d_0(v)$	The number of 0-edges incident with vertex v, called the *0-degree* of v.
$N_0(S, G)$	For $S \subseteq V$, $N_0(S, G) = \{w \notin S : \exists\, v \in S$ such that $(v, w) \in E_0\}$. ($N_0(S, G)$ is the neighborhood of S using only 0-edges).
a.a.	The random graph $G_{n,c/n}$ has property Q *almost always* (a.a.) if $\lim_{n\to\infty} P(G_{n,c/n}$ has property $Q) = 1$.
$Exp(X)$	The expected value of the random variable X.
$tv(\delta, n)$	Tour value found by algorithm searching to a depth of δ.
$\overline{X}$	An upper bound for the sequence of random variables X_n, where $\lim_{n\to\infty} P(X_n \leq \overline{X}) = 1$.
$\underline{X}$	A lower bound for the sequence of random variables X_n, where $\lim_{n\to\infty} P(X_n \geq \underline{X}) = 1$.
γ	A function that is $O(\log n)$
ϵ_c	A small constant such that $\epsilon_c > (\frac{c}{\gamma} + \frac{c}{10} e^{-2c/3})$

3 Algorithm BTS

Our algorithm uses sequential edge exchanges to search for lower valued tours. A sequential edge exchange is simply the exchanging of one edge for another adjacent to the first. This will become clear with the introduction of the algorithm, and graphs which we use to track the different steps of the algorithm.

Recall that we are given a complete graph G on n vertices, numbered $1, 2, \ldots, n$, with edge weights 0 and 1. An initial tour is given which may be chosen in any way independent of our algorithm. Let $T_0 = \{e \in E$: e is a 0-edge in this initial tour$\}$. Call this the first *Base Set*. In general, we have this definition.

Definition: A *Base Set* is the collection of 0-edges that is a subset of a tour. The Base Set which is input to iteration k will be denoted by T_{k-1}.

During the first iteration of BTS, the algorithm tries to find a set of 0-edges of size $|T_0| + 1 = |T_1|$. The set must be a subset of some tour, so it may contain no cycles and no vertices may have 0-degree ≥ 3. (When we refer to no cycles, we always mean no cycles of length $< n$. If we have a cycle of length n, BTS has found a tour of length 0). This leads to the following definition.

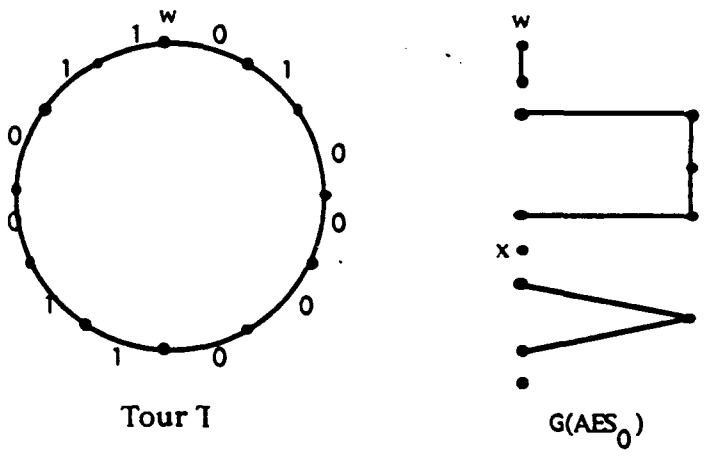

Figure 1: Tour and underlying Base Set. Edge weights are on the tour T.

Definition: An *Acceptable Edge Set* (AES) is a subset of E_0 such that the graph induced by E_0 has no cycles of length $< n$ and no vertices of 0-degree > 2. Usually the AES will be subscripted to denote how deep we have searched within an iteration to find it.

Notice that by definition, an AES will be a subset of a tour. In fact, our base sets will be acceptable edge sets. The algorithm will only keep track of this type of edge set. During a general iteration k, BTS is given a Base Set (which must be an acceptable edge set), T_{k-1}, and it tries to find an acceptable edge set, T_k, of size $1 + |T_{k-1}|$ (which will then be a new Base Set).

Define the graph, $G(AES_0)$, (which is dependent on the initial tour BTS will try to improve upon) as follows. This graph has the same vertex set as $G_{n,c/n}$ and edge set $AES_0 \equiv T_0$. In Figure 1, the first graph shows the initial tour T. The second shows the graph $G(AES_0)$. A general stage of the algorithm will consist of either adding an edge to this set, or performing one sequential edge exchange. The first operation finds the existence of a lower valued tour. (For example, if edge (x, w) is a 0-edge in Figure 1, we may add it to the set.) The second gives a new acceptable edge set with cardinality $|AES_0|$. This set must also be a subset of some tour. This set is denoted AES_1 since it is one sequential edge exchange away from the Base Set AES_0.

Figure 2 represents a series of sequential edge exchanges that result in a lower valued tour. Each graph is based on a different AES, and the sets are related as follows: $AES_{d+1} = AES_d \cup (x_d, y_d) \backslash (y_d, x_{d+1})$ d=0,1, and $AES_3 = AES_2 \cup (x_2, y_2)$. The edges in the sequential exchanges form a path. Look at the final AES. Since this set is also a subset of a tour, no edge needs to be removed.

To help in future analysis, we need to further categorize the edges in the sequential exchanges.

Definition: An *alternating path (ap)* corresponding to acceptable edge set AES_δ is a path with 2δ 0-edges. Every edge on the path is part of at least one sequential edge exchange used to find some $AES_d, 1 \leq d \leq \delta$. The path alternates between edges BTS adds to an AES, and those it removes. The path begins at a vertex which has degree less than two in

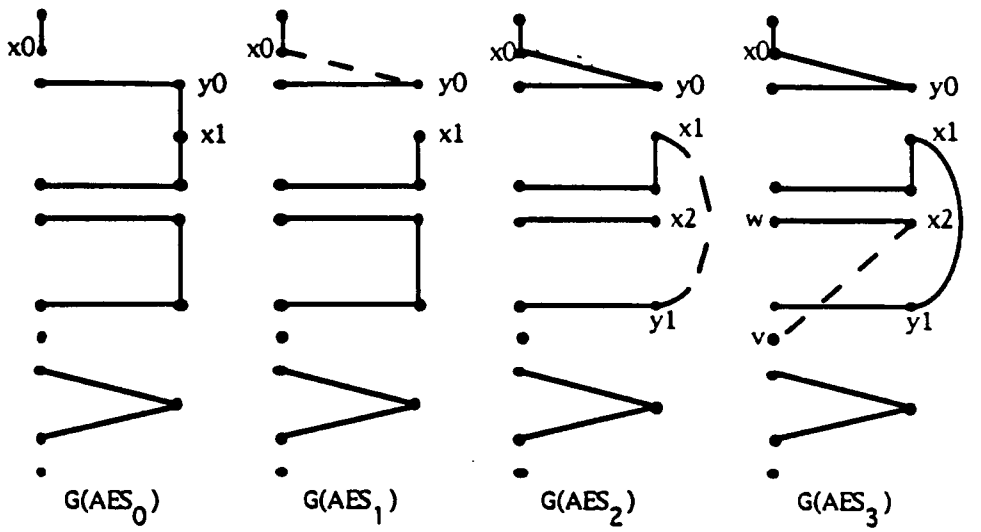

Figure 2: Graphs of acceptable edge sets, AES_3 implies a lower valued tour.

the current Base Set.

In Figure 2 an *ap* corresponding to AES_2 has vertices $(x_0, y_0, x_1, y_1, x_2)$.

Definition: A *proper alternating path (pap)* corresponding to acceptable edge set AES_δ is an alternating path extended by one 0-edge not in AES_δ. Notice that a *pap* completes an iteration since it finds a larger acceptable edge set.

In Figure 2, the edge set $\{(x_0, y_0), (y_0, x_1), (x_1, y_1), (y_1, x_2), (x_2, y_2)\}$ forms a *pap*.

We can see that finding a *pap* ensures us of finding a lower valued tour. BTS will search for these *paps*. Notice that our representation of the AES graphs (See Figures 1, 2) have the property that vertices on the left side of the graphs are incident to at least one 1-edge in the Base Set. A *pap* must start and end with this set of vertices. Thus we leave them on the same side of the graph until we find a new and larger Base Set. When a *pap* is found, BTS resets the final AES to be AES_0 (and we update our graph), and begins again.

Searching for *paps* is the main idea of the algorithm BTS. This searching is done by the procedure Search. Before presenting Search, we define the variables it uses.

Definition: Given any Base Set, AP_0 is the set of vertices which have 0-degree ≤ 1. (These vertices may start the *aps*).

Definition: Given an AES and a vertex v of 0-degree ≤ 1, we define a set of vertices, Y . A vertex $y \in Y$, is such that $y \in N_0(v)$ and $(v, y) \notin AES$. (The 0-edges $(v, y) : y \in Y$ may be added to an *ap* ending at v).

Definition: Given an AES and a vertex $y \in Y$, we define a set of vertices X . A vertex $x \in X$ is such that $x \in N_0(y)$ and $(y, x) \in AES$. (The 0-edges $(y, x) : x \in X$ are candidates to be removed from the AES and thus added to the *ap*).

Definition: δ_{max} is the maximum depth to which BTS will search.

The recursive procedure Search is presented next. In a call to the procedure Search, δ designates the current depth of the search, AES is the current edge set, and v is the vertex the algorithm will search from to extend the current *ap*. Initially $v = 0$, since we have no designated starting vertex.

Procedure Search(δ, AES, v)
Begin
If $v = 0$ and $\delta = 0$ (starting vertex is not designated)
Then Compute AP_0 (vertices which may start *aps*)
  For $v \in AP_0$:
    Search$(0, AES, v)$ (search from all vertices of 0-degree < 2)
Else ($v \neq 0$, and vertex to search from is specified)
  While $\delta < \delta_{max}$
    Compute Y (set of next possible vertices on path)
    For $y \in Y$:
      If $AES \cup (v, y)$ is an acceptable edge set
      Then **Search**$(0, AES \cup (v, y), 0)$ (Start new iteration)
      Else
        Compute X (set of next possible vertices on path)
        For $x \in X$:
          Search$(\delta + 1, AES \cup (v, y) \backslash (y, x), x)$
      Endif
Endif
End

The flowchart (Figure 3) shows a nonrecursive representation of the algorithm. A new iteration occurs when a larger Base Set is found, and the algorithm in effect starts over.

Figure 2 shows an iteration of BTS. The iteration stops when BTS finds a larger Base Set. In the procedure, this occurs when the call Search$(0, AES \cup (v, y), 0)$ is made. The vertices on the *pap* are labelled x_d and y_d, according to the depth at which the edges on the path were added or removed. The different graphs in the figure show the progression of acceptable edge sets seen until BTS finds a new Base Set, which is AES_3

4 Implementation of BTS

The algorithm BTS simply makes selected calls to the procedure Search. The time required by Search increases greatly as δ_{max} increases. When the current Base Set has a small cardinality, we can reason that we can probably find a *pap* using a small δ_{max}. This notion stems from the fact that a small sized Base Set means there are many vertices which may be used to start and end *paps*. For this reason as well as the time requirements, we will implement BTS using an increasing sequence of δ_{max}.

BTS will call Search using $\delta_{max} = 1$ first. When the procedure stops, we increase δ_{max} to 2 and keep going. Additional increases in the maximum depth of Search are made until BTS has searched to the maximum depth we are willing to allow.

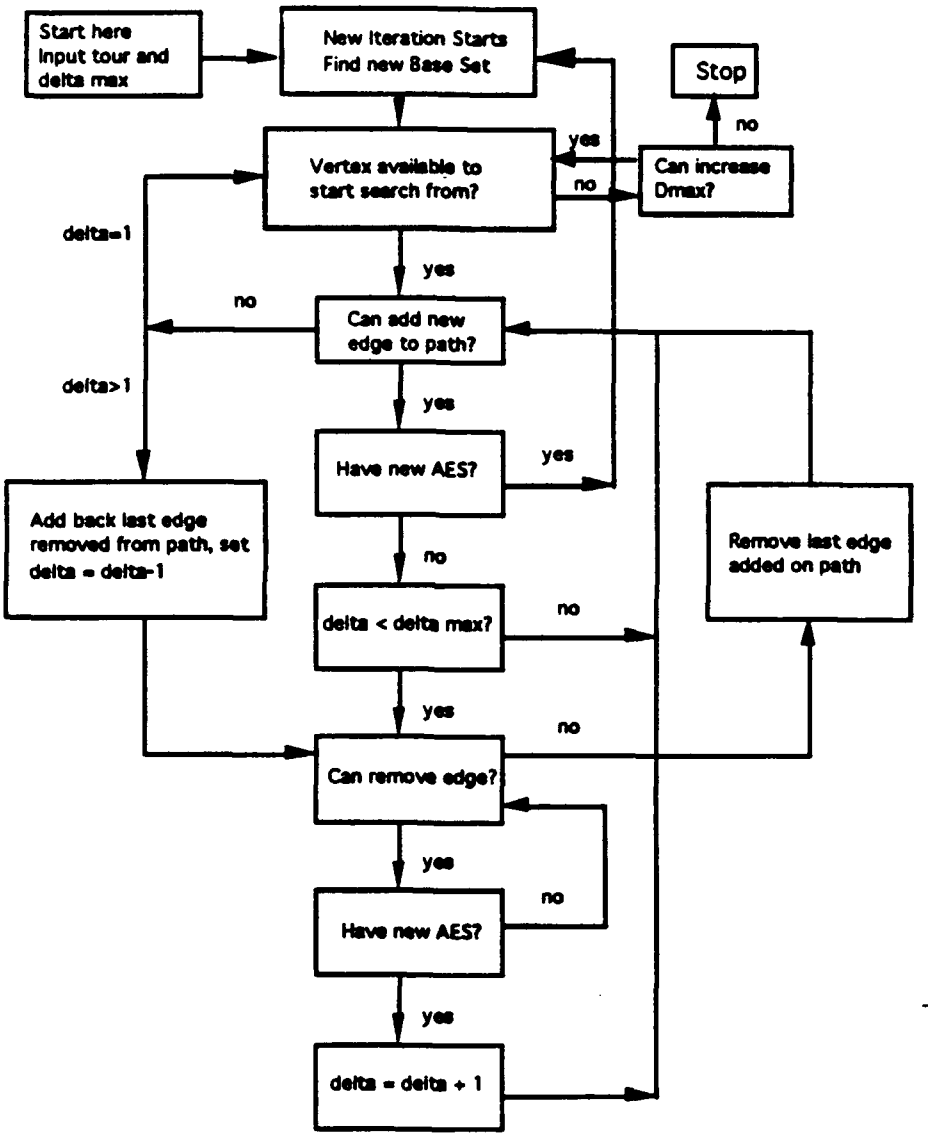

Figure 3: Nonrecursive Flowchart of Algorithm BTS

In addition, BTS makes one more change in its calls to Search. Whenever BTS reaches an AES at depth $\lfloor \delta_{max}/2 \rfloor$, it will start a new *ap*. This means BTS want to use this AES as a Base Set, and have the option of starting new *aps* from any vertex of 0-degree ≤ 1 in the AES. We can accomplish this modification by slightly changing the procedure Search. Let SplitSearch(δ, AES, v, d) be the modified procedure Search.

Procedure SplitSearch(δ, AES, v, d)
Begin
If $v = 0$ and ($\delta = 0$ or $\delta = \lfloor d/2 \rfloor$)
(starting vertex is not designated)
Then Compute AP_0 (vertices which may start *aps*)
  For $v \in AP_0$:
    SplitSearch$(0, AES, v, d)$ (search from all vertices of 0-degree < 2)
Else ($v \neq 0$, and vertex to search from is specified)
  While $\delta < d$
    Compute Y (set of next possible vertices on path)
    For $y \in Y$:
      If $AES \cup (v, y)$ is an acceptable edge set
      Then **SplitSearch**$(0, AES \cup (v, y), 0, d)$ (Start new iteration)
      Else
        Compute X (set of next possible vertices on path)
        For $x \in X$:
          If $\delta = \lfloor d/2 \rfloor - 1$
          Then SplitSearch$(\delta + 1, AES \cup (v, y) \backslash (y, x), 0, d)$
          Else SplitSearch$(\delta + 1, AES \cup (v, y) \backslash (y, x), x, d)$
          Endif
      Endif
Endif
End

The algorithm BTS may now be written as follows:

Algorithm BTS
Input the initial tour and calculate T_0. Input δ_{max}, the maximum depth to which BTS may search.
  Do for $d = 1$ to δ_{max}
  SplitSearch$(0, AES, v, d)$
  Continue
Output the current AES. Find a tour containing the AES and output it as well.
STOP

Note that ANY tour that BTS finds containing a certain AES has value $n - |AES|$. This is true since any extra 0-edge which could be found by hooking the 0-edge components together would have been found as a *pap* of length 1.

We will forgo the explanation of the need for the modification to Search until the next chapter, which presents an asymptotic analysis of BTS. The modification has intuitive rationale and is also needed to vastly improve the bounds the analysis gives.

The time required for BTS is dominated by the time it takes to search for new acceptable edge sets. Angluin and Valiant [1] have developed a data structure that enables us to find a new acceptable edge set (adding an edge and removing an edge) in time $O(\log n)$. Since the maximum 0-degree of a vertex is $4 \log n$ (a.a.), we have $O(n^2(\log n)^{\delta_{max}})$ possible AES operations (a.a.) per iteration. The n^2 comes from the fact that we rest rt the searching after depth $\delta_{max}/2$. Thus the overall time requirement is a.a. $O(n^3(4 \log n)^{\delta_{max}} \log n)$. This shows that even for large δ_{max}, say $\delta_{max} \leq \frac{\log n}{\log \log n}$, the time requirement is quite small (note that $\log n^{\frac{\log n}{\log \log n}} = n$).

Unfortunately, given as much time as we choose (δ_{max} very very large), we still may not be able to find an optimal tour. This follows because of the sequential ordering that we talked about earlier, that is, there may be tours that we cannot find using sequential edge exchanges (see [9]). However, we have evidence already that our algorithm will perform well. Our algorithm is is a generalization of HAM and of the LK algorithm. HAM has nice theoretical backings. The class of graphs for which HAM is analyzed has the property that the proportion of instances for which HAM finds a Hamiltonian Cycle approaches the probability a Hamiltonian Cycle exists - given a random graph $G_{n,m}$ on n vertices and with m edges:

$$lim_{n \to \infty} Pr(G_{n,m} \text{is Hamiltonian}) = \begin{cases} 0 & \text{if } c_n \to -\infty \\ e^{-e^{-2c}} & \text{if } c_n \to c \\ 1 & \text{if } c_n \to \infty. \end{cases}$$

This implies that sequential edge exchanges will probably take care of most of the searching we need to do. The LK algorithm has nice empirical backing, so the algorithm should do well even on small problems. This also demonstrates the power of sequential exchanging.

5 Analysis of BTS

For the analysis, it is convenient to think of BTS in terms of separate iterations. For the analysis, we assume that when BTS finds a larger Base Set, it throws out all other information and starts over again.

This section gives an asymptotic performance analysis of BTS. Earlier we defined the parameter for our Bernoulli distribution to be $c^* < c/n < 1.1 \log n$. (If $c > 1.1 \log n$ we know that a 0-length tour can be found a.a.) Many of the graph properties which we will use call for c^* to be restricted. We will need to have $c^* > 10$ always, and will at times restrict c^* to be much larger (> 400).

5.1 Analysis Overview

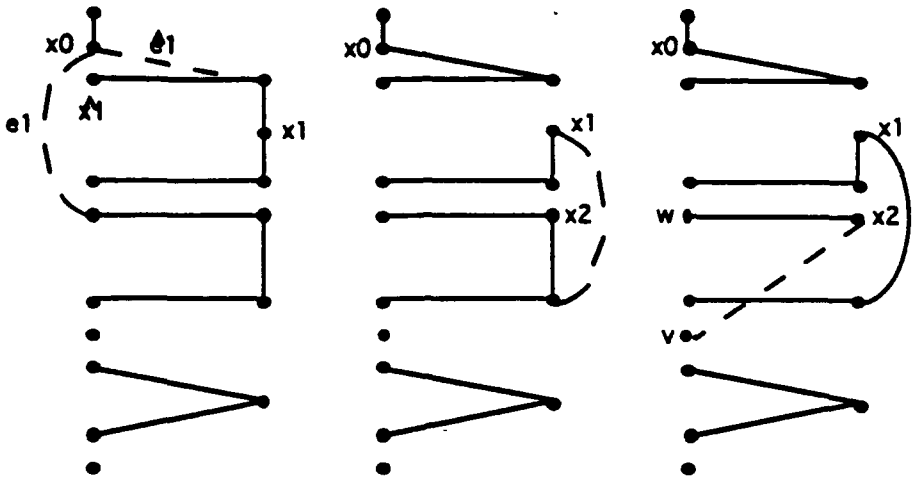

Figure 4: Graphs of acceptable edge sets, some edges BTS may add are dashed

Here we give an outline of the steps we use to analyze BTS. We will give an intuitive explanation of these steps.

Look at Figure 4. Starting at vertex x_0, BTS looks for a new edge to add. If BTS adds an edge such as e_1, a new Base Set is formed and BTS starts a new iteration. If an edge such as $\hat{e_1}$ is added, BTS must remove an edge to form a new AES. Although BTS did not find a larger Base Set, it gave itself new opportunites by finding x_1. BTS may now search from vertex x_1. In this stage, BTS finds an edge e_2 which implies a new vertex x_2, and at the next stage BTS finally finds a new Base Set.

A definition will help us here.

Definition: An *ap-reachable vertex* is any vertex used by BTS from which to search for a new edge to enter an acceptable edge set. (These are the x vertices in the *aps*.) If this vertex is found at depth δ during some iteration, then it will be referred to as a δ-reachable vertex. The collection of all vertices which are δ-reachable will be denoted by AP_δ.

It is clear that the "deeper" BTS is allowed to search (the larger the δ_{max}), the better the chance it has to find a new (larger) Base Set. This is a direct result of BTS being able to find more vertices which it sees as δ-reachable vertices. Thus the **first** step in the analysis of BTS will be to find a lower bound for the number of these vertices (as a function of δ_{max}). (To make sure we find all vertices, we assume that BTS fails to find a larger base set during the current iteration.)

Consider one new vertex that is discovered by BTS as an δ-reachable vertex. (See vertex x_2 in Figure 4, for example). [This new vertex x_d could form a new Base Set with nearly all vertices adjacent to at most 1 edge in the current AES, if a 0-edge connected them (for example 0-edge (x_2, v) implies a new Base Set, while 0-edge (x_2, w) does not).] If BTS fails during some iteration κ, many edges it sees of the above form (like (x_2, v) or e_1), must have weight 1, or BTS would use these edges to form a new Base Set. Let us call these edges, *necessary 1-edges*, for they are necessary to BTS' failure at iteration κ. This is made formal

in the following definition.

Definition: A *necessary 1-edge* is a 1-edge, e, for which there exists an *ap* found by BTS that extended by e (changed into a 0-edge) would form a *pap*. That is, it is a 1-edge which if changed to a 0-edge would imply a *pap*.

Definition: Ψ_δ is the set of necessary 1-edges found at depth δ.

The **second** step in this analysis is to find a lower bound for the number of necessary 1-edges BTS sees at a certain iteration, assuming it fails to improve during this iteration. (For ease of presentation, we refer to BTS failing *during* an iteration if it is the final iteration of the algorithm.)

The **third** step of the analysis is to find an expression for the number of 0-edges BTS sees in the Base Sets T_k, $0 \leq k \leq \kappa - 1$. If we are to find an expression for the probability BTS fails during iteration κ, we must have information about the number of 0-edges BTS has already seen. In fact, there is an inverse relationship between these two items. The more 0-edges BTS sees, the higher the probability BTS has seen all 0-edges, so the probability it fails must increase.

Now, in order to put all of this together and analyze BTS, we need one final bit of information. Given that BTS fails during iteration κ, we look for a set of 0-edges that do not influence the outcome of BTS. All that is necessary to find such a set is to find 0-edges that are not in any of the Base Sets T_k, $0 \leq k \leq \kappa - 1$. Any of these edges could be changed to 1-edges, and BTS would still see all of the same Base Sets. This is the **fourth** step of our analysis.

Here is a brief idea of how we put these steps together. Given a particular instance of G, randomly choose a subset, $\mathcal{X}$, of the 0-edges of G. Now run BTS on $G_{\mathcal{X}}$ where the edges in $\mathcal{X}$ have been changed to 1-edges. Suppose that BTS run on this new graph fails during iteration κ. Assume that $\mathcal{X}$ did *not* influence the outcome of BTS in iterations $1, 2, ..\kappa - 1$. Since BTS has failed at iteration κ, BTS has seen many, many necessary 1-edges during this iteration. If there are enough necessary 1-edges, there will be a high probability that at least one of them is in $\mathcal{X}$. Thus BTS will probably *not* fail on G, even though it did fail when a few 0-edges of G were changed to 1-edges.

For reference, here are the steps we will take:
Assume BTS fails during iteration κ:
(1) Find a lower bound for the number of vertices discovered as a δ-reachable vertex (for $0 \leq \delta \leq \delta_{max} - 1$).
(2) Given (1), find a lower bound for the number of necessary 1-edges BTS sees during iteration κ.
(3) Find an upper bound for the number of 0-edges seen in the Base Sets T_k, $0 \leq k \leq \kappa - 1$.
(4) Establish the existence of $\mathcal{X}$, a set of 0-edges that does not influence the outcome of BTS.
(5) Put everything together for the final result.

5.2 Graph Properties

Here we collect some graph properties which we will need for the five steps of our analysis.

Lemma 5.1 *[3] Let $G = G_{n,c/n}$ and let vertex v be 'small' if $d_0(v) \leq c/10$ and 'large' otherwise. Let SMALL, LARGE be the sets of small and large vertices respectively. (A small vertex has few incident 0-edges)*

Let $W(E_0) = W_1(E_0) \cup W_2(E_0) \cup W_3(E_0) \cup W_4(E_0)$, where
$W_k(E_0) = \{v : v$ is small and there exists a small w such that v and w are joined by a path of length k comprised only of 0-edges $\}$. (v=w is allowed for k=3,4).

Let $\ell \geq 7$ be fixed. Then for $c > 20(\ell+1)\log(\ell+1)$, G satisfies the following (a.a.):

$|\{v \in V : d_0(v) \leq c/10 + 1\}| \leq ne^{-2c/3}$

$d_0(v) \leq 4\log n$ *for all* $v \in V$;

$|W(E_0)| \leq c^4 e^{-4c/3} n$;

$\emptyset \neq S \subseteq V$, $|S| \leq n/2\ell$ *and* $S \subseteq LARGE$ *implies* $|N_0(S)| \geq \ell|S|$;

Next are some new lemmas.

Lemma 5.2 *Let $S_i = \{v :$ vertex v has 0-degree $i\}$, then G a.a. satisfies the following:*

(5.2.1) $||S_0| - n(1-c/n)^{n-1}| \leq n^{1/2}\log n$

(5.2.2) $||S_1| - (n-1)c(1-c/n)^{n-2}| \leq n^{1/2}\log n$

(5.2.3) $|(|S_0| + |S_1|) - n(1-c/n)^{n-1} - (n-1)c(1-c/n)^{n-2}| \leq n^{1/2}\log n$

This lemma is proved using Chebyshev's inequality, see appendix for proof.

Lemma 5.3 *Given $G_{n,c/n} = (V,E)$, let $S \subseteq V$. Let $E_0(S) = \{e = (v,w) : v,w \in S$ and (v,w) is a 0-edge $\}$. Then for all $\emptyset \neq S \subseteq V$, G a.a. satisfies the following:*

If $|S| = n/j$, where $j + j\log j < c/10$ then $|E_0(S)| \geq Exp[|E_0(S)|]/5$

To prove this lemma use the Markov Inequality, $P(|\alpha| \geq 1) \leq Exp(|\alpha|)/1$. Let α =number of sets of size $|S|$ such that $|E_0(S)| < Exp[|E_0(S)|]/5$. See appendix for full proof.

5.3 Main Steps of Analysis

The next lemma concerns the number of *alternating paths* of a given length that BTS will find during one iteration. We will try to bound the number of vertices in AP_δ, (for each δ, $0 \leq \delta \leq \delta_{max} - 1$). This will accomplish the **first** step in our analysis.

Before we start, recall that $T_{\kappa-1}$ was the Base Set that BTS failed to improve upon (since we assume that BTS fails during iteration κ).

Recall if $v \in AP_\delta$, then it is x_δ in some AES_δ. Notice that $|AP_0|$ is the set of vertices with 0-degree < 2 in the Base Set $T_{\kappa-1}$.

We can find many bounds for the size of AP_δ, the following lemma gives one. It is worth noting that this lemma really bounds the number of x_δ seen on *one* alternating path. Recall that we let BTS restart at depth $\lfloor \delta_{max}/2 \rfloor$. To avoid confusion now we will assume that $\delta \leq \lfloor \delta_{max}/2 \rfloor$. Then, we will address this restarting in the second step of the analysis and will take into account the added bonus of restarting our *aps* at $\delta = \lfloor \delta_{max}/2 \rfloor$. So for now, think of BTS searching on *one* alternating path where $\delta < \lfloor \delta_{max}/2 \rfloor$.

Lemma 5.4 *Suppose that BTS* terminates *during iteration* κ *on* $G_{n,c/n}$, *and that* $\delta \leq \lfloor \delta_{max}/2 \rfloor$. *Let* $\ell \geq 9$ *be fixed and* $c/20 \geq (\ell+1)\log(\ell+1)$ *(since we use Lemma 5.1). The following hold a.a.:*

If $|AP_0| \geq \ell n e^{-2c/3}$ *then:*

$$|AP_{\delta+1}| \geq (\frac{\ell-5}{2})|AP_\delta| + \frac{1}{2}ne^{-2c/3} \geq (\frac{\ell-5}{2})^{(\delta+1)}|AP_0| + \frac{1}{2}ne^{-2c/3}$$

as long as $|AP_\delta| \leq \frac{n}{2\ell}$. *If* $|AP_\delta| > \frac{n}{2\ell}$ *then* $|AP_{\delta+1}| \geq (\frac{\ell-5}{2})(\frac{n}{2\ell})$.

Proof:
The proof of this lemma uses techniques from [4] [5].

Consider edges $(x_\delta, y) \in E_0(G)$, where $x_\delta \in AP_\delta$ and $y \in N_0(x_\delta, G)$, and $\delta < \lfloor \delta_{max}/2 \rfloor$. We keep δ small since BTS in effect spreads itself out up to a depth of $\lfloor \delta_{max}/2 \rfloor$, and then starts over again. Assume $|AP_\delta| \leq n/2\ell$.

If $(x_\delta, y) \notin AES_\delta$ then BTS creates a new AES by adding this edge and removing a different one incident with y. Such an edge must exist since BTS fails during iteration κ, and actually two edges may exist. In the latter case we will choose only one of these edges. The edge we choose is called $(y, x_{\delta+1})$ and we will refer to the vertex $x_{\delta+1}$ as $x_{\delta+1}(x_\delta, y)$. Notice that $x_{\delta+1}$ is in $AES_{\delta+1}$, a new acceptable edge set. If $(x_\delta, y) \in AES_\delta$, then let $x_{\delta+1}(x_\delta, y) = x_\delta$.

Notice that

$$|AP_{\delta+1}| \geq |\{x_{\delta+1} = x_{\delta+1}(x_\delta, y) : x_\delta \in AP_\delta, y \in N_0(AP_\delta, G), \text{ and } (x_\delta, y) \notin AES_\delta\}|$$

$$\geq |\{x_{\delta+1} = x_{\delta+1}(x_\delta, y) : x_\delta \in (AP_\delta \cap LARGE),\ y \in N_0(AP_\delta \cap LARGE)\}| - |(AP_\delta \cap LARGE)|$$

The final inequality is true since we need to subtract at most ONE edge incident to each x_δ that could be in AES_δ. (There is at most one of these edges per x_δ).

Next, we find an expression for the number of vertices called y:

$$|N_0(AP_\delta \cap LARGE)| \leq |\{y \in N_0(AP_\delta \cap LARGE) : (y, x_{\delta+1}(x_\delta, y)) \notin T_{\kappa-1}\}| + 2|\{x_{\delta+1} : x_\delta \in (AP_\delta \cap LARGE), y \in N_0(AP_\delta \cap LARGE), (y, x_{\delta+1}) \in T_{\kappa-1}\}|$$

Clearly,

$$\begin{aligned} &|\{x_{\delta+1} = x_{\delta+1}(x_\delta, y) : x_\delta \in (AP_\delta \cap LARGE),\ y \in N_0(AP_\delta \cap LARGE)\}| \\ &\geq |\{x_{\delta+1} : x_\delta \in (AP_\delta \cap LARGE), y \in N_0(AP_\delta \cap LARGE), (y, x_{\delta+1}(x_\delta, y)) \in T_{\kappa-1}\}| \\ &\geq \frac{|N_0(AP_\delta \cap LARGE)|}{2} - \frac{|\{y \in N_0(AP_\delta \cap LARGE) : (y, x_{\delta+1}(x_\delta, y)) \notin T_{\kappa-1}\}|}{2} \end{aligned}$$

Therefore by substituting into the first expression we find that:

$$|AP_{\delta+1}| \geq \frac{|N_0(AP_\delta \cap LARGE)|}{2} - |(AP_\delta \cap LARGE)| - \frac{|\{y \in N_0(AP_\delta \cap LARGE) : (y, x_{\delta+1}(x_\delta, y)) \notin T_{\kappa-1}\}|}{2}$$

Let $Y_d = \{v : v \in N_0(AP_\delta \cap LARGE)$ and vertex v was seen as y_d during iteration $\kappa\}$. Note that Y_d is in general a proper subset of $N_0(AP_\delta \cap LARGE)$.
We also have that:

$$\begin{aligned} &|\{y \in N_0(AP_\delta \cap LARGE) : (y, x_{\delta+1}(y)) \notin T_{\kappa-1}\}| \\ &\leq |\{v : v \in AP_d, \text{or } v \in Y_d,\ 0 \leq d < \delta\}| \\ &\leq ne^{-2c/3} + |\{v : v \in \cup_{d=0}^{\delta-1}[(AP_d \cap LARGE) \cup (Y_d \cap LARGE)]\}| \end{aligned}$$

The first inequality holds since the left hand side is certainly less than the total number of vertices incident to all edges added during iteration κ. (Edge $(y_\delta, x_{\delta+1})$ must have been added to some AES during this iteration). The second inequality holds since we have at most $ne^{-2c/3}$ small vertices.

Next notice that any large vertex in AP_d will also be in AP_{d+2}. Notice that y_d will be $\in Y_{d'}$, for all $d' \geq d$. Thus we can complete our inequality with:

$$\begin{aligned}
&|\{y \in N_0(AP_\delta \cap LARGE) : (y, x_{\delta+1}(y)) \notin T_{\kappa-1}\}| \\
&\leq ne^{-2c/3} + |Y_{\delta-1}| + |AP_{\delta-1} \cap LARGE| + |AP_{\delta-2} \cap LARGE| \\
&\leq ne^{-2c/3} + 2|AP_\delta| + |AP_{\delta-1}| + |AP_{\delta-2}|
\end{aligned}$$

The final inequality holds since there are at most 2 $y_d/nd(y_d, x_{d+1}) \in T_{\kappa-1}$.

Plugging in all of this lets us see that:

$$\begin{aligned}
|AP_{\delta+1}| &\geq \frac{|N_0(AP_\delta \cap LARGE)|}{2} - \frac{ne^{-2c/3}}{2} - |AP_\delta| - \frac{|AP_{\delta-1}|}{2} \\
&\quad - \frac{|AP_{\delta-2}|}{2} - |AP_\delta \cap LARGE| \\
&\geq \frac{\ell-2}{2}(|AP_\delta| - ne^{-2c/3}) - \frac{ne^{-2c/3}}{2} - |AP_\delta| - \frac{|AP_{\delta-1}|}{2} - \frac{|AP_{\delta-2}|}{2} \\
&\geq \frac{\ell-4}{2}|AP_\delta| - \frac{\ell-1}{2}ne^{-2c/3} - \frac{|AP_{\delta-1}|}{2} - \frac{|AP_{\delta-2}|}{2}
\end{aligned}$$

These final steps use Lemma 5.1. We finish the 2 parts of the lemma by using induction. Assume $|AP_0| \geq \ell ne^{-2c/3}$.

$$\begin{aligned}
|AP_1| &\geq \frac{\ell-4}{2}|AP_0| - \frac{\ell-1}{2}ne^{-2c/3} \\
&\geq \frac{\ell-5}{2}|AP_0| + \frac{1}{2}ne^{-2c/3} \\
|AP_2| &\geq \frac{\ell-4}{2}|AP_1| - \frac{\ell-1}{2}ne^{-2c/3} - \frac{|AP_0|}{2} \\
&= \frac{\ell-5}{2}|AP_1| + \frac{1}{2}[\frac{\ell-5}{2}|AP_0| - |AP_0| - (\ell-1)ne^{-2c/3}] \\
&\geq \frac{\ell-5}{2}|AP_1| + \frac{1}{2}ne^{-2c/3}
\end{aligned}$$

Assume true for $|AP_\delta| \leq n/2\ell$, show true for $|AP_{\delta+1}|$.

$$\begin{aligned}
|AP_{\delta+1}| &\geq \frac{\ell-4}{2}|AP_\delta| - \frac{\ell-1}{2}ne^{-2c/3} - \frac{|AP_{\delta-1}|}{2} - \frac{|AP_{\delta-2}|}{2} \\
&\geq \frac{\ell-5}{2}|AP_\delta| + \frac{1}{2}[\frac{\ell-5}{2}|AP_{\delta-1}| - |AP_{\delta-1}| - |AP_{\delta-2}| - (\ell-1)ne^{-2c/3}]
\end{aligned}$$

$$\geq \frac{\ell-5}{2}|AP_\delta| + \frac{1}{2}ne^{-2c/3} + \frac{1}{2}[|AP_{\delta-2}| - \ell ne^{-2c/3}]$$
$$\geq (\frac{\ell-5}{2})^{(\delta+1)}|AP_0| + \frac{1}{2}ne^{-2c/3}$$

□

Assume BTS fails during iteration κ on the graph G. Step two of our analysis is to find an expression for the number of 1-edges BTS finds that would create a *pap* if any one of the 1-edges were changed to a 0-edge. This is the set of necessary 1-edges which we defined earlier.

We have come to the part of this analysis where we make use of the fact that BTS restarts itself at depth $\lfloor \delta_{max}/2 \rfloor$. From Lemma 5.4, we know that the largest sized AP_δ we can guarantee is $(\frac{\ell-5}{2})(\frac{n}{2\ell})$, for any δ. Earlier we saw that most edges having one vertex in AP_δ and one in AP_0 were necessary. This implies we can guarantee almost $[(\frac{\ell-5}{2})(\frac{n}{2\ell})][\ell ne^{-2c/3}]$ necessary edges (may have to have δ very large). However, this turns out to be not nearly enough necessary edges to give us good bounds for guaranteed tour values that BTS will find. This is especially true if c is large.

However, if we restart the searches after $\lfloor \delta_{max}/2 \rfloor$, then we may use all vertices in $AP_{\lfloor \delta_{max}/2 \rfloor}$ to finish *paps*. This is true since every vertex in this set has 0-degree < 2 in some $AES_{\lfloor \delta_{max}/2 \rfloor}$. This AES will be used as a Base Set. Thus we have two groups of vertices, $AP_{\lfloor \delta_{max}/2 \rfloor}$ and $AP_{\delta_{max}}$, which we may use to guarantee necessary edges. If BTS fails then it will see $\approx \frac{1}{2}[(\frac{\ell-5}{2})(\frac{n}{2\ell})]^2$ necessary 1-edges. The formal statement and proof follow.

Lemma 5.5 *Given the conditions of Lemma 5.4, the following holds a.a.:*

$$|\Psi_2| \geq min[\frac{\ell-5}{2}|AP_0|^2\frac{1}{2}, \frac{\ell-5}{2}\frac{n}{2\ell}(|AP_0|-2)\frac{1}{2}]$$
$$|\Psi_{\delta_{max}}| \geq \min[(\frac{\ell-5}{2})^{\delta_{max}-1}|AP_0|^2\frac{1}{2}, (\frac{\ell-5}{2})^2(\frac{n}{2\ell})^2\frac{1}{2}]$$

for δ_{max} greater than 2

Proof:
The general idea of this proof is to show that even if BTS fails, it will still find many pairs of *aps*. Each *ap* pair implies an edge which could hook them together and in most cases, this edge is necessary. Thus we need only use Lemma 5.4 to bound the number of *ap* pairs that BTS will find, (equivalently we find vertices in $AP_{\lfloor \delta_{max}/2 \rfloor}$ and $AP_{\delta_{max}}$, and then show which of these vertex pairs a new Base Set when they are joined together by a 0-edge. Thus, we are counting necessary edges between pairs of *aps*.

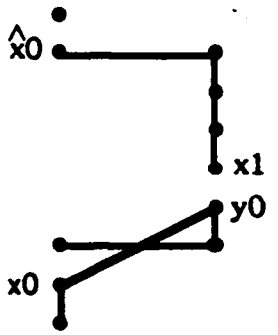

Figure 5: Graph of AES_1

We will rename the vertices BTS sees to emphasize the fact that it will find two *aps* and then try and hook them together. Denote the vertices on the two *aps* by x_0, y_0, $x_1, y_1, x_2, y_2, x_3, y_3$,...$x_{\lfloor \delta_{max}/2 \rfloor}$ (first *ap*) and $\hat{x}_0, \hat{y}_0$ ..., $\hat{x}_1, \hat{y}_1$, ..., $\hat{x}_{\lfloor \delta_{max}/2 \rfloor - 1}$ (second *ap*). $\hat{x}_0$ is the first vertex on the second *ap*, and is chosen by BTS at depth $\lfloor \delta_{max}/2 \rfloor$.

We first prove the bound for $\delta_{max} = 2$. The first step in this proof is to bound the number of endpoints of *aps* of length one. By Lemma 5.4 we have:

$$|AP_1| \geq \min[\ |AP_0|(\frac{\ell-5}{2}) + \frac{1}{2}ne^{-2c/3},\ (\frac{n}{2\ell})(\frac{\ell-5}{2})]$$

Next look at any $x_1 \in AP_1$ and its corresponding AES_1. (See Figure 5). If BTS fails at an iteration with $\delta_{max} = 2$, then x_1 is adjacent to at least $|AP_0| - 2$ necessary 1-edges. The only 0-edges adjacent to a vertex in AP_0 which may not be necessary are $(x_0, x_1(x_0, y_0))$ and $(\hat{x}_0, x_1)$, where the latter edge may form a cycle in $AES_1 \cup (\hat{x}_0, x_1)$ (as in Figure 5).

Thus each x_1 vertex $\Rightarrow (|AP_0| - 2)$ necessary 1-edges, and this gives a bound for the total number of necessary 1-edges BTS will find (a.a.). This bound is:

$$\begin{aligned} |\Psi_2| &\geq \min[(\frac{\ell-5}{2})(|AP_0| + \frac{1}{2}ne^{-2c/3})(|AP_0| - 2)\frac{1}{2}\ ,\ (\frac{n}{2\ell})(\frac{\ell-5}{2})(|AP_0| - 2)\frac{1}{2}] \\ &\geq \min[(\frac{\ell-5}{2})|AP_0|^2\frac{1}{2}\ ,\ (\frac{n}{2\ell})(\frac{\ell-5}{2})(|AP_0| - 2)\frac{1}{2}] \end{aligned}$$

(The 1/2 in the expressions accounts for counting *paps* up to twice, once in either direction.)

Next, we prove the general case for δ_{max} odd and ≥ 3. Recall, we assume BTS fails during iteration κ. To prove the bound for the number of necessary 1-edges BTS will find (a.a.), we look for necessary edges between the two possible *aps*.

As before, if we could find a lower bound for the guaranteed number of these *ap* pairs, and consequently a lower bound for the number of edges used to hook these pairs together, we could find a lower bound for the number of necessary 1-edges BTS will find. (The necessary 1-edges are the edges which can be used to hook together the two *aps*.)

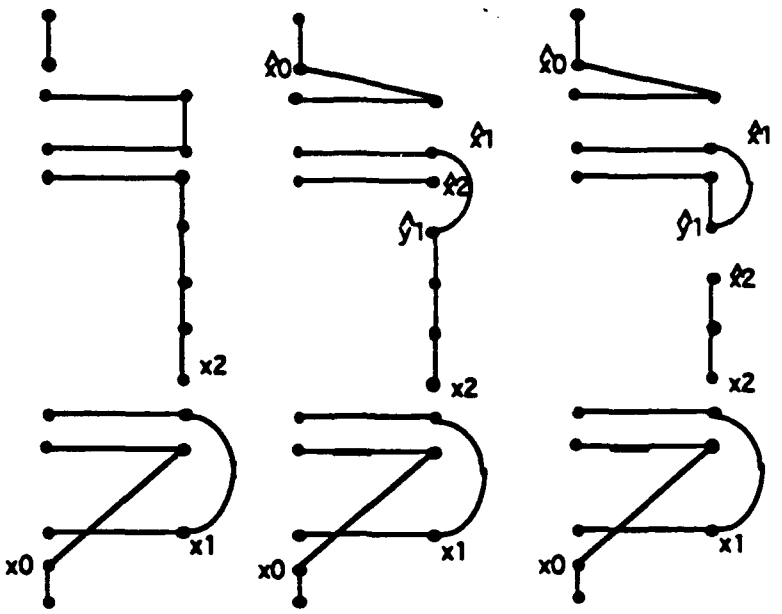

Figure 6: Graphs of AES_2 and 2 possible AES_4, where $\delta_{max} = 5$.

We proceed as in the case where $\delta_{max} = 2$. First we bound the number of endpoints of *aps* of length $\lfloor \delta_{max}/2 \rfloor$. This is equivalent to bounding the size of $|AP_{\lfloor \delta_{max}/2 \rfloor}|$. Again, by Lemma 5.4 this is as follows:

$$|AP_{\lfloor \delta_{max}/2 \rfloor}| \geq \min[(\frac{n}{2\ell})(\frac{\ell-5}{2}),\ |AP_0|(\frac{\ell-5}{2})^{\lfloor \delta_{max}/2 \rfloor}]$$

We next look at any vertex denoted by an $x_{\lfloor \delta_{max}/2 \rfloor}$ and its corresponding $AES_{\lfloor \delta_{max}/2 \rfloor}$. (More than one AES may exist: choose one.) Using $AES_{\lfloor \delta_{max}/2 \rfloor}$ as a Base Set, we want to bound the number of endpoints which imply necessary edges in $AES_{\delta_{max}-1}$ sets. We will do this by counting the endpoints of only certain *aps*. (We continue to use the new notation for *ap* vertices defined at the start of this proof).

The following characterizes the second *aps* that we will count. Given a fixed *ap* of length $(\delta_{max} - 1)/2$ (since δ_{max} is odd, $\lfloor \delta_{max}/2 \rfloor = (\delta_{max} - 1)/2$) and its acceptable edge set $AES_{(\delta_{max}-1)/2}$, any *ap* found from this Base Set will imply a necessary edge of the form $(x_{(\delta_{max}-1)/2}, \hat{x}_{(\delta_{max}-1)/2})$ if the following are satisfied:

1. $\hat{x}_0 \neq x_0$
2. No $\hat{x}_d$ is in same component with $x_{(\delta_{max}-1)/2}$.

It is easy to show that these conditions are sufficient. Condition 1 is needed since x_0 may have 0-degree 2 and thus may not be used to start a new *ap*. If the second condition is satisfied no cycles will be created if the necessary edge is added as a 0-edge. (See Figure 5)

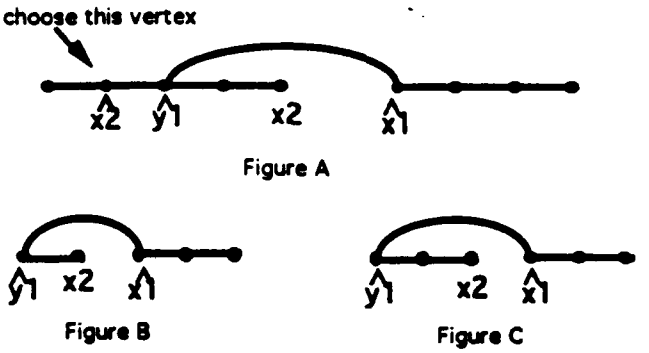

Figure 7: Adding edges in second *ap*, $\delta_{max} = 5$, $x_{(\delta-1)/2} = x_2$

Given any $x_{\lfloor \delta_{max}/2 \rfloor}$ and a corresponding $AES_{\lfloor \delta_{max}/2 \rfloor}$, recall that in the proof of Lemma 5.4 we used the following idea: first, for any edge added (here we add $(\hat{x}_d, \hat{y}_d)$) during an iteration of BTS, we take into account the removal of only *one* of two possible edges. The edge we choose to account for is called $(\hat{y}_d, \hat{x}_{d+1})$ and we refer to the vertex $\hat{x}_{d+1}$ as $\hat{x}_{d+1}(\hat{x}_d, \hat{y}_d)$. Though we may have two choices for the vertex $\hat{x}_{d+1}$ (neither imply cycles), we choose only one to *count*. In this analysis, we stipulate that if $\hat{y}_d$ is in the same component as the fixed vertex $x_{(\delta_{max}-1)/2}$ (in the current AES), we choose $\hat{x}_{d+1}$ so that it will not be in the same component as $x_{(\delta_{max}-1)/2}$ in the *next* AES. (See Figure 6A, our fixed vertex is $x_2 = x_{(\delta_{max}-1)/2}$). Figure 6B shows the correxct choice for $\hat{x}_2$, Figure 6C the incorrect choice for $\hat{x}_2$.)

As long as $\hat{x}_0$ is not initially in the same component with $x_{(\delta_{max}-1)/2}$, we can always make this choice. No cycles will ever be formed. The proof and bounds will still be valid. See Figure 7 for the three possible results of adding the next edge to the second *ap*. 7A and 7B illustrate how to choose the next vertex. 7C forms a *pap*, which cannot happen since BTS fails during this iteration.

We are basically done. We need just note that for any fixed *ap* and its corresponding AES, we can start the second *ap* from $(|AP_0| - 2)$ vertices. (This will follow the same reasoning as in the case for $\delta_{max} = 2$). We may not be able to start the second *ap* from x_0 or from a vertex in the same component with $x_{\lfloor \delta_{max}/2 \rfloor}$. (See Figure 5). As in the bound for the number of endpoints of the first *aps*, we use Lemma 5.4. The number of vertices BTS will find that we have called $\hat{x}_{(\delta_{max}-1)/2}$ is no less than:

$$\min[(\frac{\ell-5}{2})^{(\delta_{max}-1)/2}(|AP_0| - 2), (\frac{n}{2\ell})(\frac{\ell-5}{2})]$$

We need now only multiply the two bounds together, and divide by two. This will account for counting the edges at most twice, as the endpoint pairs can be used in both *aps*. Note that $|AP_1|(|AP_0| - 2) \geq \frac{(\ell-5)}{2}|AP_0|^2$. We have also have accounted for this in the final expression of this lemma. This concludes the proof for δ_{max} odd.

The expression is proved for δ_{max} even and ≥ 4 in exactly the same way. The only difference is that the second *ap* is one edge shorter than the first.
□

We now start the third step of our analysis, which is to find an upper bound for the number of 0-edges seen in the Base Sets T_k, $0 \leq k \leq \kappa - 1$.

Now, instead of thinking about BTS in terms of iterations, it will behoove us to think about sets of iterations that use the same δ_{max}. We will do this by induction on δ_{max}. Later, we will analyze BTS by showing how long BTS can continue (a.a.) until it needs to increase δ_{max}.

Definition: $B(G,\kappa) = \{\text{1-edges in initial tour} \cup (\cup_{k=0}^{\kappa-1} T_k)\}$. $B(G,\kappa)$ is the union of the initial tour and all of the Base Sets seen through iteration $\kappa - 1$.

We give a lemma that enables us to bound $|B(G,\kappa)|$.

Recall that $tv(\delta,n)$ is the value of the tour found when BTS can search to a depth of δ. Also, BTS will first look for *paps* of length 1, then those of length ≤ 2, etc. Thus you may also think of $tv(\delta,n)$ as being the tour value where BTS must increase its depth from δ to $\delta+1$ in order to continue its search for a lower valued tour.

Lemma 5.6 *Given BTS is at iteration κ and is using δ_{max} on graph G, then the following holds:*

$$|B(G,\kappa)| \leq 2n + \sum_{\delta=1}^{\delta_{max}-1} tv(\delta,n)$$

Proof:

BTS runs at $\delta = 1$ until it cannot add any more *paps* of length one. The underlying tour has value $tv(1,n)$. It then runs using $\delta \leq 2$, until it stops (with underlying tour of value $tv(2,n)$). BTS keeps increasing depth until it reaches δ_{max}.

The initial tour uses n edges. How many additional edges are seen in Base Sets when BTS searches for *paps* of length one? There will be at most

$$(n - tv(1,n)) \leq n$$

Thus if $\delta_{max} = 1$, we have

$$|B(G,\kappa)| \leq n + (n - tv(1,n)) \leq 2n$$

Increasing δ_{max} to 2 increases the number of edges seen in Base Sets by at most $2(tv(1,n) - tv(2,n))$. Thus we have:

$$\begin{aligned} |B(G,\kappa)| &\leq n + (n - tv(1,n)) + 2(tv(1,n) - tv(2,n)) \\ &\leq 2n + tv(1,n) - 2tv(2,n) \\ &\leq 2n + tv(1,n) \end{aligned}$$

Note that the final bound holds for any iteration which uses $\delta_{max} = 2$. You can see this easily generalizes to any δ_{max}. We have:

$$\begin{aligned} |B(G,\kappa)| &\leq n + (n - tv(1,n)) + 2(tv(1,n) - tv(2,n)) \\ &\quad +... + \kappa(tv(\kappa - 1,n) - tv(\kappa,n)) \\ &\leq 2n + tv(1,n) + tv(2,n) + ... + tv(\kappa - 1,n) - \kappa(tv(\kappa,n)) \\ &\leq 2n + tv(1,n) + ... + tv(\kappa - 1,n) \end{aligned}$$

□

We now come to the **fourth** step in our analysis. We need to come up with an expression for the number of edges that do not influence BTS until the iteration κ during which BTS fails. (Notice that since we assume that BTS terminates during iteration κ, BTS returns a tour value of $n - |T_{\kappa-1}|$ and finds a tour containing the 0-edges in $T_{\kappa-1}$.)

Before we can go any further, we need to *define* our set of noninfluential edges.

Definition: $\mathcal{X} \subseteq E_0$ is called *deletable* if:
(1) No edge of $\mathcal{X}$ is incident with a vertex of 0-degree $\leq c/10$.
(2) The edges in $\mathcal{X}$ form a matching in $G_{n,c/n}$.
(3) $\mathcal{X} \cap B(G,\kappa) = \emptyset$.

The deletable set idea is used in [2] and [4]; we have altered it for our purposes. It should be clear that the third property of $\mathcal{X}$ keeps it from influencing BTS before iteration κ. As we saw earlier, this does not mean that BTS will run exactly the same when edges in $\mathcal{X}$ become 1-edges. BTS may not see all previous *aps* now. However, BTS will see exactly the same Base Sets as before.

Lemma 5.7 *If* $|B(G,\kappa)| \leq nc/30$ *and* $c > 95$, *then there a.a. exists a deletable set of size* $\frac{\gamma \log n}{2}$, *where* γ *is any function that is* $O(\log n)$.

We prove this in 4 steps.
1. $B(G,\kappa)$ can contain all edges incident to no more than than $20(|B(G,\kappa)| - n)/(c - 10)$ large vertices.
2. $\mathcal{X}$ has at least $n - ne^{-2c/3} - 20(|B(G,\kappa)| - n)/(c - 10) = \alpha$ large vertices left to choose edges from.
3. If $|B(G,\kappa)| \leq nc/30$, the number of edges left to choose from is $\geq \alpha(\alpha - 1)c/10n$
4. Find a matching from vertices in 2, using Lemma 5.1 and Lemma 5.3.

5.4 Putting it all together

Next we use all of the previously found information to analyze BTS. First a quick preview: in order to prove that BTS can find a tour of a certain value (a.a.), we prove it does not fail during iterations corresponding to larger tour values (a.a.). The proofs will center upon the deletable set idea. We will generate a set $\mathcal{X}$ and let $G_{\mathcal{X}}$ be the graph G, except that edges in $\mathcal{X}$ have been changed to 1-edges.

The general idea is as follows: suppose BTS fails during iteration κ, and $\mathcal{X}$ is deletable. It follows that BTS fails on $G_{\mathcal{X}}$. Futhermore, BTS will see the same Base Sets $(T_k, 0 \leq k \leq \kappa - 1)$ on G *and* $G_{\mathcal{X}}$. This is true because BTS is a deterministic algorithm. Given the same realization and starting tour, BTS will always perform identically. We will use Lemma 5.5 to guarantee that if we run BTS on $G_{\mathcal{X}}$ the algorithm will see many, many necessary 1-edges in $G_{\mathcal{X}}$. Under certain conditions, the probability that none of these necessary 1-edges are in $\mathcal{X}$ will be so small that $P(BTS$ fails during iteration $\kappa) \rightarrow 0$ as $n \rightarrow \infty$. Of course we know by Lemma 5.2 that BTS must fail by a certain iteration. However, we want to know how small of a tour value BTS can find, as a function of δ_{max}.

Important Note: The reader may have noticed that we have been sneaky. The reader may think, if we want to use those lemmas, that we need to go back and change them to hold true for $G_{\mathcal{X}}$, instead of G. Well, actually the lemmas that may be affected (5.4 and 5.5) are true for $G_{\mathcal{X}}$ as well as G. Since we wanted to do the calculations only once, we did them for $G_{\mathcal{X}}$. How can we see this? Notice that we defined "small" vertices as having $\leq c/10$ incident 0-edges, but we used a bound for vertices of 0-degree $\leq c/10 + 1$ (see Lemma 5.1). So when we subtracted off the small vertices, we included vertices that may have become small due to changing the weights of edges in $\mathcal{X}$.

After generating our edge weights on $G_{n,c/n}$, we *randomly and independently* color the edges of E_0 green with probability $\gamma \log n/cn$. Call the green edges E_{og}. We will want to see if these edges can be a deletable set. Note that

$$Exp[|E_{og}|] = \frac{n(n-1)}{2}\frac{c}{n}\frac{\gamma \log n}{cn} \approx (\gamma \log n)/2$$

The following method is analogous to that in Frieze [4]. First, we must restrict our Bernoulli distribution. We let $c > 20(\ell + 1)\log(\ell + 1)$, where $\ell \geq 9$. Let it be given that BTS has reached iteration κ.

We need the following conditions:

1. $|\{v \in V : d_0(v) \leq c/10 + 1\}| \leq ne^{-2c/3}$

2. $d_0(v) \leq 4 \log n$ for all $v \in V$

3. $|\Psi_2| \geq \min[\frac{\ell-5}{2}|AP_0|^2\frac{1}{2}, \frac{\ell-5}{2}\frac{n}{2\ell}(|AP_0|-2)\frac{1}{2}$

$|\Psi_{\delta_{max}}| \geq \min[(\frac{\ell-5}{2})^{\delta_{max}-1}|AP_0|^2\frac{1}{2}, (\frac{\ell-5}{2})^2(\frac{n}{2\ell})^2\frac{1}{2}$
for δ_{max} greater than 2

4. $|B(G,\kappa)| \leq 2n + \sum_{\delta=1}^{\delta_{max}-1} tv(\delta,n)$

5. If $|B(G,\kappa)| \leq nc/30$, then there a.a. exists a deletable set of size $\frac{\gamma \log n}{2}$, where $\gamma = O(\log n)$.

Let L be the event that conditions 1-5 hold. Note that $P(L) = 1 - o(1)$.
Define the following events:

$\mathcal{E}_1 = [(BTS \text{ fails during iteration } k \text{ on } G) \cap L]$

$\mathcal{E}_2 = [\mathcal{E}_1 \cap (\mathcal{X} = E_{og} \text{ is deletable})\ ]$

We will investigate

$$P(\mathcal{E}_1) = \frac{P(\mathcal{E}_2)}{P(\mathcal{E}_2|\mathcal{E}_1)}$$

We first investigate $P(\mathcal{E}_2|\mathcal{E}_1)$. We will use $\overline{|B(G,\kappa)|}$, which by definition is an upper bound for $|B(G,\kappa)|$ which holds a.a.

Lemma 5.8

$$P(\mathcal{E}_2|\mathcal{E}_1) \geq (1-o(1))(1-\frac{\gamma \log n}{cn})^{\overline{|B(G,\kappa)|}+(c/10)ne^{-2c/3}}$$

Proof: If BTS fails on G, $\mathcal{X}$ need only satisfy its 3 requirements up to iteration κ.

We have that:

$$\begin{aligned} P(\mathcal{E}_2|\mathcal{E}_1) &= P(\mathcal{X} \cap B(G,\kappa) = \emptyset,\ \mathcal{X} \cap E_0(SMALL) = \emptyset, \\ &\quad \mathcal{X} \text{ forms a matching }) \\ &= P(\mathcal{X} \text{ forms a matching } |\mathcal{X} \cap (B(G,\kappa) \cup E_0(SMALL)) = \emptyset) \\ &\quad * P(\mathcal{X} \cap (B(G,\kappa) \cup E_0(SMALL)) = \emptyset) \end{aligned}$$

We'll show:
(1) $P(\mathcal{X}$ forms a matching $|\mathcal{X} \cap (B(G,\kappa) \cup E_0(SMALL)) = \emptyset) = (1-o(1))$

(2) $P(\mathcal{X} \cap (B(G,\kappa) \cup E_0(SMALL)) = \emptyset) \geq (1 - \frac{\gamma \log n}{cn})^{\overline{|B(G,\kappa)|}+(c/10)ne^{-2c/3}}$

to prove the Lemma.

To show (1), let $\alpha =$ [number of vertices with at least 2 incident edges in $\mathcal{X}$]. Next use the Markov Inequality and show that $Exp(\alpha) = o(1/n)$.

Item (2) is easily derived by noticing that there are $\leq (c/10)ne^{-2c/3}$ 0-edges (attached to small nodes) that may not be colored green. By definition $\overline{|B(G,\kappa)|}$ is an upper bound for $|B(G,\kappa)|$. Later we will find a valid value for $\overline{|B(G,\kappa)|}$

(Full proofs may be found in the appendix.) □

Now we will investigate $\mathcal{E}_2$.

Lemma 5.9

$$P(\mathcal{E}_2) \leq [(1 - \frac{(\gamma \log n)(c)}{(cn)(n)})]^{\underline{|\Psi_{\delta_{max}}|}}$$

To prove, follow these simple steps:

$\mathcal{E}_2 =$ [$\mathcal{E}_1 \cap \mathcal{X} = E_{og}$ is deletable]

$\mathcal{E}_2 \Longrightarrow$ [BTS fails on $G_{\mathcal{X}}$ during iteration κ AND $\mathcal{X} \cap \Psi_{\delta_{max}}(G_{\mathcal{X}}) = \emptyset$ AND L occurs]

thus
$\mathcal{E}_2 \subseteq$ [BTS fails on $G_{\mathcal{X}}$ during iteration $\kappa \bigcap (\mathcal{X} \cap \Psi_{\delta_{max}}(G_{\mathcal{X}}) = \emptyset) \bigcap L$]

and clearly
$P(\mathcal{E}_2) \leq P(\mathcal{X} \cap \Psi_{\delta_{max}} = \emptyset)$

Now, recall that the edges $e \in \mathcal{X}$ were colored independently. By definition $\underline{|\Psi_{\delta_{max}}|}$ is a lower bound for $|\Psi_{\delta_{max}}|$ Thus we have:

$$P(\mathcal{E}_2) \leq [P(e_\psi \notin \mathcal{X})]^{\underline{|\Psi_{\delta_{max}}|}}$$

where $e_\psi \in \Psi_{\delta_{max}}$. Hence,

$$P(\mathcal{E}_2) \leq [(1 - \frac{(\gamma \log n)(c)}{(cn)(n)})]^{\underline{|\Psi_{\delta_{max}}|}}$$

□

Putting the two Lemmas together:

Theorem 5.1 *Given a random graph $G_{n,c/n}$. The following holds a.a.:*

$$P(BTS\ fails\ during\ iteration\ \kappa\ |BTS\ succeeds\ up\ to\ iteration\ \kappa-1) =$$

$$P(\mathcal{E}_1) = \frac{P(\mathcal{E}_2)}{P(\mathcal{E}_2|\mathcal{E}_1)} \leq \frac{(1-\frac{\gamma\log n}{n^2})^{|\Psi_{\delta_{max}}|}}{(1-o(1))(1-\frac{\gamma\log n}{cn})^{|B(G,\kappa)|+(c/10)ne^{-2c/3}}}$$

where γ is any function that is $O(\log n)$, and we must have $|B(G,\kappa)| \leq nc/30$.

Next we look for relationships between $|B(G,\kappa)|$ and $|\Psi_{\delta_{max}}|$. Given BTS is at iteration κ, we want to find the smallest δ_{max} we can use and still have the probability BTS fails go to 0. By keeping δ_{max} as small as possible, $|B(G,\kappa)|$ is smaller and we can proceed to do more iterations before $|B(G,\kappa)| > nc/30$.

Please keep in mind that the probability
$P(\mathcal{E}_1) = P((BTS$ fails during iteration κ on $G) \cap L)$
is conditioned on BTS running to iteration κ. However, we would like to have an unconditioned bound for this probability. We may use conditional probability to see that:

$$\begin{aligned} &P(\text{BTS fails during iteration } \kappa) \\ &= \sum_{k=1}^{\kappa} P(\text{BTS fails during iteration } \kappa|\text{BTS gets to iteration } \kappa) \\ &\quad *P(\text{BTS gets to iteration } \kappa) \end{aligned}$$

Since BTS can have up to n iterations, we need $P(\mathcal{E}_1) \leq o(1/n)$. It will be sufficient to show that (a.a.)

$$e^{\frac{-\gamma\log n}{n^2}|\Psi_{\delta_{max}}|+\frac{\gamma\log n}{cn}(|B(G,\kappa)|+(c/10)ne^{-2c/3})} = o(1/n)$$

Why? Let

$$A = (1 - \frac{\gamma \log n}{n^2})^{\underline{|\Psi_{\delta_{max}}|}} \quad , \quad B = e^{\frac{-\gamma \log n}{n^2}\underline{|\Psi_{\delta_{max}}|}}$$

$$C = (1 - \frac{\gamma \log n}{cn})^{\overline{|B(G,\kappa)|}+(c/10)ne^{-2c/3}} \quad , \quad D = e^{-\frac{\gamma \log n}{cn}(\overline{|B(G,\kappa)|}+(c/10)ne^{-2c/3})}$$

$$P(\mathcal{E}_1 \leq (1+o(1))\frac{A}{C} = (1+o(1))\frac{A}{B} * \frac{B}{D} * \frac{D}{C}$$

Now $A/B \leq 1$ for all n. We'll show that $\lim_{n\to\infty} \frac{D}{C} = 1$. Let $\alpha n = (\overline{|B(G,\kappa)|} + (c/10)ne^{-2c/3})$. We will need the following property: if $t < .43$, then $-\log t(1-t) < (t + t^2/2 + t^3/2)$.

$$\frac{D}{C} = \frac{n^{\frac{-\gamma\alpha}{c}}}{(1-\frac{\gamma \log n}{cn})^{\alpha n}}$$

$$\begin{aligned}
\lim_{n\to\infty} \frac{D}{C} &= e^{\lim_{n\to\infty}(\log D - \log C)} \\
&= e^{\lim_{n\to\infty}(-\gamma\alpha/c)\log n - \alpha n \log(1-\frac{\gamma \log n}{cn})} \\
&\leq e^{\lim_{n\to\infty}(-\gamma\alpha/c)\log n + \alpha n[\frac{\gamma \log n}{cn} + \frac{(\gamma \log n)^2}{2cn^2} + \frac{(\gamma \log n)^3}{3cn^3}]} \\
&= e^{\lim_{n\to\infty} \alpha(\frac{(\gamma \log n)^2}{2cn} + \frac{(\gamma \log n)^3}{3cn^2})} \\
&= e^0 \\
&= 1
\end{aligned}$$

So now we know that if (a.a.)

$$\frac{B}{D} = e^{\frac{-\gamma \log n}{n^2}\underline{|\Psi_{\delta_{max}}|} + \frac{\gamma \log n}{cn}(\overline{|B(G,\kappa)|}+(c/10)ne^{-2c/3})} = o(1/n)$$

BTS will find any tour values this equation implies.

This is equivalent to showing that

$$-\frac{\gamma\underline{|\Psi_{\delta_{max}}|}}{n^2} + \frac{\gamma}{cn}(\overline{|B(G,\kappa)|} + (c/10)ne^{-2c/3}) < -1$$

We can see from the Lemma 5.4, that γ may be any function that is $O(\log n)$. From now on, we assume it is in fact a $\Theta(\log n)$.

Before we get to the next theorem, here is a corollary to demostrate the implications of the theorem. Recall that $tv(\delta_{max}, n)$ represents the value of the smallest tour BTS finds using δ_{max}. Let $TV(\delta_{max}, n) = \sum_{\delta=0}^{\delta_{max}} \overline{tv}(\delta, n)$, where we define $\overline{tv}(0, n) := (2 + \epsilon_c)\, n$.

Corollary 5.1 *Given a random graph $G_{n,c/n}$ and $c \geq 461$, (as $c \geq 20(\ell+1)\log(\ell+1)$ and $\ell \geq 9$). If*

$$|B(G, \kappa+1)| < \frac{nc}{30} \quad \textit{and} \quad TV(\delta_{max}, n) < \frac{cn}{2}(\frac{\ell-5}{4\ell})^2$$

then the following holds (a.a.):

$$\overline{tv}(\delta+1, n) \leq \rho_\delta \overline{tv}(\delta, n)$$

and ρ_δ converges to $(\frac{2}{\ell-5})^{1/2} + \epsilon$, where ϵ is $O(e^{-c/3})$.

The restrictions on $|B(G, \kappa+1)|$ and $\overline{tv}(\delta, n)$ will be justified in the proof of the following theorem. As we increase δ_{max}, our guaranteed tour values decrease geometrically. The general theorem is a bit more messy, but you should be able to see the pattern. The corollary is proved after the proof of the theorem.

We have already seen from Lemma 5.2 that the optimal tour value, tv^*, can be bounded as follows:

$$tv^* \geq n(1-c/n)^{n-1} + \frac{(n-1)c}{2}(1-c/n)^{n-2} - n^{1/2}\log n$$

See page 32 for a discussion of the relationship between this lower bound and the upper bounds given in the previous theorem.

Theorem 5.2 *Given a random graph $G_{n,c/n}$. The following hold a.a.:*

(1) If $\delta_{max} = 1$ and $c \geq 10$

$$tv(1, n) \leq \overline{tv}(1, n) = \frac{n}{j}, \textit{ where } j + j\log j < \frac{c}{10}$$

(2) Let $\ell \geq 9$ be fixed, where $(\ell+1)\log(\ell+1) < c/20$. (Thus $c \geq 461$). Then

$$tv(2, n) \leq \overline{tv}(2, n) = [\frac{2TV(1, n)n}{c}\frac{2}{\ell-5}]^{1/2}$$

$$tv(\delta_{max}, n) \leq \overline{tv}(\delta_{max}, n) = [\frac{2TV(\delta_{max}-1, n)n}{c}(\frac{2}{\ell-5})^{\delta_{max}-1}]^{1/2}$$

for $\delta_{max} \geq 3$ *and as long as* $\overline{tv}(\delta_{max}, n) > \ell n e^{-2c/3}$, *and* $TV(\delta_{max}, n) < \frac{cn}{2}(\frac{\ell-5}{4\ell})^2$.

Proof:
We prove the case for $\delta_{max} = 1$ first. Suppose that BTS is starting iteration κ, and $[n - |T_{\kappa-1}|] > n/j$. The number of vertices with 0-degree < 2 in $T_{\kappa-1}$ is $= |AP_0| \geq [n - |T_{\kappa-1}| + 1]$.

By Lemma 5.3, $|E_0(AP_0)| \geq Exp[|E_0(AP_0)|]/5 = \frac{c|AP_0|(|AP_0|-1)}{10n} > \frac{cn}{10j^2}$.

The number of 0-edges in $[E_0(AP_0) \cap T_{k-1}] = |AP_0| - (n - |T_{\kappa-1}|) - g$, where $g =$ the number of components of size ≥ 3 vertices in $T_{\kappa-1}$. ($g =$ number of 0-edges that would form cycles in AP_0, see Figure 4 for reference.) Thus the number of edges in $E_0(AP_0)$ that can form *paps* of length 1 is at least

$$\begin{aligned} \frac{c|AP_0|(|AP_0|-1)}{10n} - [|AP_0| - (n - |T_{\kappa-1}|) - g] - g &> \frac{c|AP_0|(|AP_0|-1)}{10n} - |AP_0| \\ &\geq \frac{c(|AP_0|-1)}{10j} - |AP_0| \\ &> 1 \end{aligned}$$

since $c/10 \geq j + j\log j$.

Notice we did not need the idea of a deletable set in this part of the proof; that will come next. We have found an upper bound for $tv(1,n)$, the smallest tour value BTS can find (a.a.) using *paps* of length 1 only. We can now plug this bound and others which we have computed into the inequality from Theorem 5.1 to find an upper bound for $tv(2,n)$. This is the smallest tour value found by using *paps* of length 1 and 2 only. In order to keep $|B(G,\kappa)|$ small, we have made sure that BTS uses $\delta_{max} = 1$ while $|T_{\kappa-1}| > n/j$, where n/j is as in the above lemma. Once we reach the minimum n/j, BTS will use $\delta_{max} = 2$.

To prove the bound for $tv(2,n)$, we want to find valid bounds for $|\Psi_{\delta_{max}}|$ and $|B(G,\kappa)|$ such that:

$$e^{\frac{-\gamma\log n}{n^2}|\Psi_2| + \frac{\gamma\log n}{cn}(\overline{|B(G,\kappa)|} + (c/10)ne^{-2c/3})} = o(1/n)$$

or equivalently

$$n^{\frac{-\gamma}{n^2}|\Psi_2| + \frac{\gamma}{cn}(\overline{|B(G,\kappa)|} + (c/10)ne^{-2c/3})} = o(1/n)$$

It suffices to show that (a.a.)

$$\frac{-\gamma}{n^2}\underline{|\Psi_2|} + \frac{\gamma}{cn}(\overline{|B(G,\kappa)|} + (c/10)ne^{-2c/3}) < -1$$

By algebra we see that we need,

$$\underline{|\Psi_2|} > \frac{n}{c}(\overline{|B(G,\kappa)|} + \frac{cne^{-2c/3}}{10} + \frac{nc}{\gamma})$$

Now, we let $\overline{|B(G,\kappa)|} = 2 + \overline{tv}(1,n)$. Since $\epsilon_c > (\frac{c}{\gamma} + \frac{c}{10}e^{-2c/3})$ we can can solve:

$$\underline{|\Psi_2|} \geq \frac{n}{c}[2 + \epsilon_c + \frac{\overline{tv}(1,n)}{n}] = \frac{TV(1,n)}{c}$$

Recall that (a.a.):

$$|\Psi_2| \geq min[\frac{\ell-5}{2}|AP_0|^2\frac{1}{2}, \frac{\ell-5}{2}\frac{n}{2\ell}(|AP_0|-2)\frac{1}{2}]$$

where $|AP_0| \geq \ell ne^{-2c/3}$. We know that given any Base Set, the tour value it implies is $\leq |AP_0| - 1$. If we can solve

$$\min[\frac{\ell-5}{2}|AP_0|^2\frac{1}{2}, \frac{\ell-5}{2}\frac{n}{2\ell}(|AP_0|-2)\frac{1}{2}] = \frac{TV(1,n)}{c}$$

for $|AP_0|$, this will imply a tour value which can be found a.a.

After algebra:

$$min[|AP_0|^2, \frac{n}{2\ell}(|AP_0|-2)] = 2\frac{2}{\ell-5}\frac{TV(1,n)n}{c}$$

Since we are trying to minimize $|AP_0|$, we may assume that $|AP_0|^2 \leq \frac{n}{2\ell}(|AP_0|-2)$. Thus

$$\frac{n}{2\ell}(|AP_0|-2) \geq \frac{4}{\ell-5}\frac{f(1)n}{c}$$

If we know that

$$(\frac{n}{2\ell}-2)(\frac{\ell-5}{2})\frac{c}{n} \geq TV(1,n)$$

then the above equation is feasible.

Minor algebra then gives:

$$tv(2,n)+1 \leq |AP_0| = [\frac{4}{\ell-5}\frac{TV(1,n)n}{c}]^{1/2} = \overline{tv}(2,n)$$

Thus we are (a.a.) assured of BTS finding a tour of at most the above value. This gives the desired result.

To prove the rest of the bounds, assume they are true for all $\delta \leq \delta_{max}$. To show this is true for $\delta_{max}+1$ we need to find the minimum $|AP_0|$ that satifies:

$$\frac{-\gamma}{n^2}\underline{|\Psi_{\delta_{max}+1}|} + \frac{\gamma}{cn}(\overline{|B(G,\kappa)|} + (c/10)ne^{-2c/3}) < -1$$

Recall that (a.a.):

$$|\Psi_{\delta_{max}+1}| \geq \min[(\frac{\ell-5}{2})^{\delta_{max}}|AP_0|^2\frac{1}{2}, (\frac{\ell-5}{2})^2(\frac{n}{2\ell})^2\frac{1}{2}]$$

Since both terms are less than $|\Psi_{\delta_{max}+1}|$ (a.a.) and $TV(\delta_{max},n) \geq |B(G,\kappa)| + (c/10)ne^{-2c/3}) + \frac{1}{\gamma}$ we may solve for:

$$\min[(\frac{\ell-5}{2})^{\delta_{max}}|AP_0|^2\frac{1}{2}, (\frac{\ell-5}{2})^2(\frac{n}{2\ell})^2\frac{1}{2}] = \frac{TV(\delta_{max},n)\,n}{c}$$

As before, since $tv(\delta,n) < |AP_0|$, this will give us a valid bound for $tv(\delta,n)$.

As we are trying to minimize $|AP_0|$, we may assume that

$$(\frac{\ell-5}{2})^{\delta_{max}}|AP_0|^2\frac{1}{2} \leq (\frac{\ell-5}{2})^2(\frac{n}{2\ell})^2\frac{1}{2}$$

As before, this implies that $TV(\delta_{max},n) \leq \frac{cn}{2}(\frac{\ell-5}{4\ell})^2$, or else no feasible solution will exist.

Thus $tv(\delta_{max}+1,n)$ is such that:

$$tv(\delta_{max}+1,n) \leq |AP_0| - 1 \leq [\frac{2TV(\delta_{max},n)n}{c}(\frac{2}{\ell-5})^{\delta_{max}}]^{1/2} = \overline{tv}(\delta_{max}+1,n)$$

and this proves the theorem, as long as $\overline{tv}(\delta_{max}+1,n) > \ell ne^{-2c/3}$.

□

For example, if $\ell = 11$ and $c \geq 597$:

$$tv(1,n) \leq \frac{n}{15.4}\,, \quad tv(2,n) \leq \frac{n}{20.81}\,, \quad tv(3,n) \leq \frac{n}{35.64}$$

$$tv(4,n) \leq \frac{n}{61.33}\,, \quad tv(5,n) \leq \frac{n}{105.83}$$

For all $\delta \geq 5$, we have:

$$\overline{tv}(\delta+1,n) \leq .57862\,\overline{tv}(\delta,n)$$

as long as $\overline{tv}(\delta+1,n) > 11ne^{-2c/3}$, and $TV(\delta,n) < 5.5n$. The rate of decrease is already very close to $1/\sqrt{3} \approx .57735$

To prove the corollary, we need to investigate the rate of decrease of the $tv(\delta,n)$, as a function of δ. As well, we need to ensure that $|B(G,\kappa)| \leq nc/30$, and $TV(\delta,n) \leq \frac{cn}{2}(\frac{\ell-5}{4\ell})^2$.

Proof of Corollary:
We need only show that if:

$$[\frac{2TV(\delta,n)n}{c}(\frac{2}{\ell-5})^{\delta-1}]^{1/2} = \rho_\delta[\frac{2TV(\delta-1,n)n}{c}(\frac{2}{\ell-5})^{\delta-2}]^{1/2}$$

then as $\delta \to \infty$, $\rho_\delta \to (\frac{2}{\ell-5})^{1/2} + \epsilon$, where ϵ is $O(e^{-c/3})$.

The first expression can be simplified to

$$TV(\delta,n)(\frac{2}{\ell-5}) = \rho^2 TV(\delta-1,n)$$

Thus

$$\rho = (1 + \frac{\overline{tv}(\delta,n)}{TV(\delta-1,n)})^{1/2}(\frac{2}{\ell-5})^{1/2}$$

Now ρ is strictly decreasing; in the above expression, the numerator is decreasing, and the denominator increasing. Note that from the previous theorem:

$$\frac{\overline{tv}(\delta,n)}{2n\,TV(\delta-1,n)})^{1/2} \leq [\frac{1}{c}(\frac{2}{\ell-5})^{\delta-2}]^{1/2}$$

as long as $\overline{tv}(\delta,n) > \ell n e^{-2c/3}$. We have that $TV(\delta-1,n) > 2n$, so as well:

$$\frac{\overline{tv}(\delta,n)}{TV(\delta-1,n)} \leq [\frac{1}{c}(\frac{2}{\ell-5})^{\delta-2}]^{1/2}$$

for large i and n this will imply that $\overline{tv}(\delta,n) < 2\ell n e^{-2c/3}$. This proves the corollary. □

Note: Though we must always satisfy $TV(\delta,n) \leq \frac{cn}{2}(\frac{\ell-5}{4\ell})^2$ and $|B(G,\kappa)| \leq nc/30$, sometimes this is always true. At $\ell = 9$, we can see that we must have $TV(\delta,n) \leq 2.845n$, but if ℓ is only a little larger, nice things happen. For example, let $\ell = 11$ and $c > 597$ as we did before. $\ell = 11 \Rightarrow$ we must have $TV(\delta,n) \leq 5.55n$. Notice that:

$$\begin{aligned} TV(\delta,n) &= TV(0,n) + \sum_{d=1}^{\delta} \overline{tv}(d,n) \\ &\leq TV(5,n) + \sum_{d=6}^{\delta} \overline{tv}(d,n) \\ &\leq 2.16786n + \sum_{j=1}^{\infty} (.57862)^j \frac{1}{105.83} \\ &\leq 2.2n \end{aligned}$$

Thus we need not worry that $TV(\delta,n) > \frac{cn}{2}(\frac{\ell-5}{4\ell})^2$ or $|B(G,\kappa)| > nc/30$.

The first theorem gave us the generalization of the analysis to include the probability BTS can find a tour of a certain value, given δ_{max} (these are true a.a.). We need only plug in the values that we have found for $\overline{|B(G,\kappa)|}$ and $\underline{|\Psi_\delta|}$, which are calculated by using the $\overline{tv}(\delta,n)$ we have just found.

The corollary and final theorem tell us that as δ_{max} is increased, BTS's (asymptotic) guaranteed tour value will converge at a rate which approaches $2/(\ell-5)^{1/2}$, to a minimum tour value of $\ell e^{-2c/3} + 1$. However, we can find smaller guaranteed tour values by using different bounds, for example, the other bound we found implies a minimum tour value of $\frac{\ell}{\ell-8}e^{-2c/3} + 1$ (but not as fast an improvement rate). As well, this analysis rests on a neighborhood argument (which we use to bound the number of necessary edges seen by BTS) which sometimes uses only vertex sets with a large 0-degree. That is, we do parts of this analysis using vertices that have 0-degree $> c/10$. By adding in contributions from 'small' vertices we can obtain smaller guaranteed tour values for large δ. The rate of improvement that we find can still approach $2/(\ell-5)^{1/2}$, and the best tour value we can find is still $O(ne^{-2c/3})$.

Now consider these two things. First, the expected value of a randomly chosen tour is $n - c$. Even using small values of δ_{max} lets us find tours with a much smaller guaranteed tour value. Second, by looking at the minimum number of vertices of 0-degree 0 or 1 (see Lemma 5.2), we find a very quick (and loose) lower bound for the optimal tour value. This bound shows that the optimal tour value is *never* less than $n(1-c/n)^{n-1} + \frac{(n-1)c}{2}(1-c/n)^{n-2} - n^{1/2}\log n$. A much tighter bound can no doubt be found, and we think it will be much closer to $O(ne^{-2c/3})$.

Our lower bound for the optimal tour value will be closest to the upper bound for our algorithms performance (with δ_{max} large) when the above bound is close to $\ell n e^{-2c/3}$. This is smallest when c is large. For example if $c = \log n$, the difference between the upper and lower bounds is $O(n^{1/3})$ but if $c = O(1)$ then the difference is $O(n)$.

6 Modifications to BTS

6.1 Analysis of 2-opting

In the previous section we mentioned that BTS could be reformulated as a strict improvement algorithm. Here we present results which come from the modification of BTS into a strict 2-opt heuristic. The key idea in the modification is to find a way to identify which *aps* and *paps* translate into 2-changes. The full analysis can be found in [13].

Theorem 6.1 *Given a random graph $G_{n,c/n}$. If ϵ_c is a small constant such that $\epsilon_c > (\frac{c}{\gamma} + \frac{c}{10}e^{-2c/3})$ (where γ is a function of $O(\log n)$) and $c \geq 95$ (for c must satisfy requirements of Lemma 5.7 and then 2-opting can find (a.a.) a tour of length tv, such that*

$$tv \leq (\frac{2+\epsilon_c}{c})^{1/2}(2n)$$

What are the time requirements of this 2-opt heuristic? We know that we can do a 2-change in time $O(\log n)$ since this is equivalent to a sequential exchange. As stated in the last section, we know that the 0-degree of any vertex is $\leq 4 \log n$ a.a. At any iteration we may have to search from $O(n)$ vertices, and we have $O(n)$ iterations. It follows that the time requirements are $O(n^2(\log n)^2)$. Note that this is the same as the time requirements for algorithm BTS, given that $\delta_{max} = 1$.

According to our analysis, the performance of our 2-opting heuristic never surpasses that of BTS (when $\delta_{max} = 1$). Notice that 2-opting is quite tour dependent, and will perform better on tours that have 1-edges grouped together. This would imply that using 2-opting after a nearest neighbor tour construction is probably a good idea. The nearest neighbor algorithm will produce exactly the tours with which 2-opting is most successful. This suggests that a modification of our 2-opt heuristic would probably be in order. We would want to do successful 2-changes (those that lead to a lower valued tour) that keep the most number of grouped 1-edges. (However, this is beyond the scope of this paper. Perhaps we can tackle this idea in the future.)

However, given our comparison of time requirements of BTS and 2-opting, it would seem unwise to use 2-opting at any time. Even if we want to use the nearest neighbor algorithm to find an initial tour, BTS will still outperform the algorithm restricted to be a strict 2-opt procedure given the same amount of time. We suspect that BTS could also be enhanced by choosing *paps* which keep 1-edges grouped together.

6.2 Further Implementations of Procedure Search

The procedure search can be implemented on a directed or assymetric 0,1 TSP. Tour improvement algorithms like 2-opting are generally not well qualified to handle the directed TSP. Picture doing a 2-change on a (directed) tour. Up to 1/2 of the edges may change direction, and radically change the tour value. But Search can handle these problems by keeping 0-edge components in their original direction. The same kind of asymptotic analysis presented in this paper can be performed for the asymmetric Bernoulli Salesman. We need only make sure we keep components in their original direction at all times. The most restricted step will be in the choice of final edges on *paps*. The directed case may also be analyzed in this manner if there are enough 0-edges present in the graph.

We also stated that Search may be implemented as a strict improvement algorithm. The current theorem (for symmetric salesman) still holds true for this implementation, though the time requirement grows. Since searching up to depth δ implies we are using up to $2\delta + 1$ edge exchanges this analysis is true for a $2\delta + 1$-opt.

In addition, we can modify Search to perform as a strict k-change algorithm. This modification is very easy to do for 2-opting or 3-opting, although a little harder for a general k-opt. We have already given results from the analysis of 2-opting. The analysis techniques can be used for any strict k-opt, however, the increasing complexity makes it hard to obtain good bounds.

7 Appendix

For the following lemmas, let $G = G_{n,c/n}$, where , $c^* \leq c \leq 1.1 \log n$ and c^* is a large constant

Proof of **Lemma 5.2**

Lemma 5.2 *Let $S_i = \{v : \text{vertex } v \text{ has 0-degree } i\}$, then G a.a. satisfies the following:*

(5.2.1) $||S_0| - n(1 - c/n)^{n-1}| \leq n^{1/2} \log n$

(5.2.2) $||S_1| - (n-1)c(1 - c/n)^{n-2}| \leq n^{1/2} \log n$

(5.2.3) $|(|S_0| + |S_1|) - n(1 - c/n)^{n-1} - (n-1)c(1 - c/n)^{n-2}| \leq n^{1/2} \log n$

Proof:
We will prove only the first expression, the proofs of the others are similar.

To establish the first expression we use Chebyshev's Inequality which tells us that:

$$P(||S_0| - Exp(|S_0|)| \geq n^{1/2} \log n) \leq \frac{Var(|S_0|)}{n(\log n)^2}$$

We have that:

$$\begin{aligned} Exp(|S_0|) &= n\binom{n-1}{0}(\frac{c}{n})^0(1-\frac{c}{n})^{n-1} \\ &= n(1-\frac{c}{n})^{n-1} \end{aligned}$$

Now we look at S_0 in terms of indicator variables:

$$|S_0| = \sum_{j=1}^{n} I_0(v_j) \text{ where } I_0(v_j) = \begin{cases} 1 & \text{if } d_0(v_j) = 0 \\ 0 & \text{if } d_0(v_j) > 0 \end{cases}$$

This allows us to find an expression for the variance of S_0.

$$\begin{aligned} Var(|S_0|) &= Exp[(\sum_{j=1}^{n} I_0(v_j))^2] - [Exp(\sum_{j=1}^{n} I_0(v_j)]^2 \\ &= nExp[I_0(v_1)^2] + 2\binom{n}{2}Exp[I_0(v_1)I_0(v_2)] - n^2Exp[I_0(v_1)]^2 \\ &= nExp[I_0(v_1)] + n(n-1)Exp[I_0(v_1)I_0(v_2)] - n^2Exp[I_0(v_1)]^2 \end{aligned}$$

We can simplify this by noting that:

$$\begin{aligned} Exp[I_0(v_1)I_0(v_2)] &= P[I_0(v_1) = I_0(v_2) = 1] \\ &= P[I_0(v_1) = 1 \mid I_0(v_2) = 1]\ P[I_0(v_1) = 1] \\ &= (1-\frac{c}{n})^{n-2}\ (1-\frac{c}{n})^{n-1} \\ &= (1-\frac{c}{n})^{2n-3} \end{aligned}$$

Plugging in:

$$\begin{aligned} Var(|S_0|) &= n(1-\frac{c}{n})^{n-1} + n(n-1)(1-\frac{c}{n})^{2n-3} - n^2(1-\frac{c}{n})^{2n-2} \\ &= n(1-\frac{c}{n})^{n-1} + n^2(1-\frac{c}{n})^{2n-2}[(1-\frac{c}{n})^{-1} - 1] - n(1-\frac{c}{n})^{2n-3} \\ &= n(1-\frac{c}{n})^{n-1} + n^2(1-\frac{c}{n})^{2n-2}[\frac{c}{n-c}] - n(1-\frac{c}{n})^{2n-3} \\ &< n(1-\frac{c}{n})^{n-1} + 2nc(1-\frac{c}{n})^{2n-2} \end{aligned}$$

Using Chebyshev, the proof of (5.2.1) is complete.
□

Proof of **Lemma 5.3**

Lemma 5.3 *Given $G_{n,c/n} = (V, E)$, let $S \subseteq V$. Let $E_0(S) = \{e = (v, w) : v, w \in S$ and (v, w) is a 0-edge $\}$. Then for all $\emptyset \neq S \subseteq V$, G a.a. satisfies the following:*

If $|S| = n/j$, where $j + j \log j < c/10$ then $|E_0(S)| \geq Exp[|E_0(S)|]/5$

Proof:
Recall the Markov Inequality, $P(|X| \geq 1) \leq Exp(|X|)/1$.

Let α =number of sets of size $|S|$ such that $|E_0(S)| < Exp[|E_0(S)|]/5$. Then

$$Exp(\alpha) = \binom{n}{|S|} \sum_{k=0}^{f} \left(\begin{matrix} \binom{|S|}{2} \\ k \end{matrix} \right) (\frac{c}{n})^k (1 - \frac{c}{n})^{\binom{|S|}{2}-k}$$

where $f = Exp[|E_0(S)|]/5 - 1$. For algebraic ease, let $f^* = (c|S|^2)/(10n)$ $(f < f^*)$.

Since the maximum value of a Binomial distribution occurs at its expected value (approximately), the maximum term in the above expression occurs at f. So we have:

$$\begin{aligned}
Exp(\alpha) &\leq \binom{n}{|S|}(f^* + 1) \left(\begin{matrix} \binom{|S|}{2} \\ f^* \end{matrix} \right) (\frac{c}{n})^{f^*} (1 - \frac{c}{n})^{\binom{|S|}{2}-f^*} \\
&\leq (\frac{ne}{|S|})^{|S|} (\frac{|S|^2 c}{10n} + 1) [\frac{(|S| - 1)10e}{2|S|}]^{\frac{|S|^2 c}{10n}} (1 - \frac{c}{n})^{\binom{|S|}{2} - \frac{c|S|^2}{10n}} \\
&\leq (\frac{ne}{|S|})^{|S|} (\frac{|S|^2 c}{10n} + 1) [\frac{(|S| - 1)10e}{2|S|}]^{\frac{|S|^2 c}{10n}} e^{\frac{-c}{n}[\frac{|S|(|S|-1)}{2} - \frac{|S|^2 c}{10n}]} \\
&\leq (\frac{ne}{|S|})^{|S|} (\frac{|S|^2 c}{10n} + 1) (5e)^{\frac{|S|^2 c}{10n}} e^{\frac{-c}{n}[\frac{|S|(|S|-1)}{2} - \frac{|S|^2 c}{10n}]} \\
&\leq (\frac{ne}{|S|})^{|S|} (\frac{|S|^2 c}{10n} + 1) e^{\frac{.27|S|^2 c}{n}} e^{\frac{-c|S|^2}{2n} + \frac{-c^2|S|^2}{10n^2} + \frac{c|S|}{2n}} \\
&\leq (\frac{ne}{|S|})^{|S|} e^{\frac{-.13c|S|^2}{n}} e^{\frac{-c^2|S|^2}{10n^2} + \frac{c|S|}{2n}} \\
&\leq (\frac{ne}{|S|})^{|S|} e^{\frac{-.1c|S|^2}{n}}
\end{aligned}$$

the final expression will go to 0 as $n \to \infty$ if $j + j \log j < c/10$.

□

Proof of **Lemma 5.7**

Lemma 5.7 *If $|B(G, \kappa)| \leq nc/30$ and $c > 95$, then there a.a. exists a deletable set of size $\frac{\gamma \log n}{2}$, where $\gamma = O(\log n)$, and c is greater than a large constant.*

Proof:
By the definition of a deletable set, all edges in $\mathcal{X}$ must be incident to large vertices only (one with 0-degree $\leq c/10$). However, $B(G,\kappa)$ may contain many edges incident to large vertices as well. How many large vertices can $B(G,\kappa)$ completely cover ($B(G,\kappa)$ contains all edges incident to how many vertices)?

1. n edges in $B(G,\kappa)$ are in the initial tour, so each vertex has at least 2 of its incident edges in $B(G,\kappa)$. Thus $B(G,\kappa)$ can use up to the following number of large vertices:

$$\frac{2(|B(G,\kappa)|-n)}{\frac{c}{10}-1} = \frac{20(|B(G,\kappa)|-n)}{c-10}$$

2. So the number of vertices left that $\mathcal{X}$ has to choose from is at least:

$$n - ne^{-2c/3} - \frac{20(|B(G,\kappa)|-n)}{c-10}$$

Assume $|B(G,\kappa)| \leq nc/30$, where $c > 10$. The the number of vertices left that $\mathcal{X}$ has to choose from is at least:

$$n - ne^{-2c/3} - \frac{2/3n(c-30)}{c-10} > n\,[\frac{1}{3} - e^{2c/3} > \frac{n}{4}]$$

3. By Lemma 5.3 if $10(4+4\log 4) < c$ (which implies $c > 95$), then the number of 0-edges present in the subgraph induced by the large vertices that $\mathcal{X}$ can choose from, is at least:

$$\frac{(n/4)^2 c}{10n} = \frac{cn}{160}$$

4. We can find an acceptable $\mathcal{X}$ by finding a matching from this subset of edges and vertices. Each vertex has $\leq 4\log n$ 0-edges incident to it, so we can greedily find a matching of the desired size. Any first edge chosen may exclude at most $(8\log n - 1)$ edges from the matching. The second edge chosen may exclude $(8\log n - 1)$ more, etc. In this way, we can easily find a matching of size at least:

$$\frac{cn/160}{8\log n} = \frac{cn}{1280\log n} \geq \frac{\gamma \log n}{2}$$

if $\gamma = O(\log n)$ and the Lemma is proved.
□

Proof of Lemma 5.8

Lemma 5.8 *Given that $G_{n,c/n}$ satisfies the conditions from Lemmas 5.1 - 5.7, and that edges in $\mathcal{X}$ were chosen independently from $E_0(G)$ with probability $(1 - \frac{\gamma \log n}{cn})$:*

$$P(\mathcal{E}_2|\mathcal{E}_1) \geq (1 - o(1))(1 - \frac{\gamma \log n}{cn})^{(c/10)ne^{-2c/3} + |\overline{B(G,\kappa)}|}$$

Proof:
We'll show:

1) $P(\mathcal{X}$ forms a matching$) \geq (1 - o(1))$

2) $P(\mathcal{X} \cap (E_0(SMALL) \cup B(G,\kappa)) = \emptyset) \geq (1 - \frac{\gamma \log n}{cn})^{(c/10)ne^{-2c/3} + |\overline{B(G,\kappa)}|}$

to prove the Lemma.

1) Let $\beta = |\{v : v$ has ≥ 2 incident edges in $\mathcal{X} | \mathcal{X} \cap (E_0(SMALL) \cup B(G,\kappa)) = \emptyset\}|$

$$\begin{aligned}
P(\beta \geq 1) &\leq \frac{Exp(\beta)}{1} \\
&= \sum_i P(\text{ vertex } v_i \text{ has } \geq 2 \text{ incident edges in } \mathcal{X} | \mathcal{X} \cap (E_0(SMALL) \cup B(G,\kappa)) = \emptyset) \\
&\leq \sum_i P(\text{ vertex } v_i \text{ has } \geq 2 \text{ incident edges in } \mathcal{X}) \\
&= n \sum_{j=2}^{n-1} \binom{n-1}{j} (\frac{\gamma \log n}{cn} \frac{c}{n})^j (1 - \frac{\gamma \log n}{cn} \frac{c}{n})^{n-1-j} \\
&= n[1 - (1 - \frac{\gamma \log n}{n^2})^{n-1} - (n-1)(\frac{\gamma \log n}{n^2}(1 - \frac{\gamma \log n}{n^2})^{n-2})] \\
&= n[1 - (1 - \frac{\gamma \log n}{n^2})^{n-2}(1 + (n-2)\frac{\gamma \log n}{n^2})]
\end{aligned}$$

Now we show that

$$(1 - \frac{\gamma \log n}{n^2})^{n-2} \geq (1 - (n-2)\frac{\gamma \log n}{n^2})$$

This is true since

$$\begin{aligned}
(1 - \frac{\gamma \log n}{n^2})^{n-2} &= 1 - (n-2)\frac{\gamma \log n}{n^2} + \binom{n-2}{2}(\frac{\gamma \log n}{n^2})^2 \\
&\quad - \binom{n-2}{3}(\frac{\gamma \log n}{n^2})^3 + \ldots (\frac{\gamma \log n}{n^2})^{n-2}
\end{aligned}$$

and for all $1 \leq j \leq n-1$:

$$\binom{n-2}{j}\left(\frac{\gamma \log n}{n^2}\right)^j > \binom{n-2}{j+1}\left(\frac{\gamma \log n}{n^2}\right)^{j+1}$$

Thus

$$\begin{aligned} P(\beta \geq 1) &\leq n\{1-[(1-(n-2)\frac{\gamma \log n}{n^2})(1+(n-2)\frac{\gamma \log n}{n^2}]\} \\ &= n[1-1+((n-2)\frac{\gamma \log n}{n^2})^2] \\ &= n[((n-2)\frac{\gamma \log n}{n^2})^2] \\ &\leq \frac{(\gamma \log n)^2}{n} \end{aligned}$$

and thus 1) is proved.

2) The edges in E_0 were colored independently, with the probability any 0-edge given by green is $(1-\frac{\gamma \log n}{cn})$. Thus:

$$\begin{aligned} P(\mathcal{X} \cap (E_0(SMALL) \cup B(G,\kappa))\emptyset) &= (1-\frac{\gamma \log n}{cn})^{\overline{|E_0(SMALL) \cup B(G,\kappa)|}} \\ &\geq (1-\frac{\gamma \log n}{cn})^{(c/10)ne^{-2c/3}+\overline{|B(G,\kappa)|}} \end{aligned}$$

since there are $\leq (c/10)ne^{-2c/3}$ *0-edges* adjacent to small vertices.

□

6 Acknowledgements

We would like to thank Craig A. Tovey for many helpful discussions about this work.
Drs. Llewellyn and Whitaker were supported in part by the Office of Naval Research (N00014-89-1658).

References

[1] Angluin D. and Valiant L., "Fast probabilistic algorithms for hamilton circuits and matchings," *Comput. System Sci.*, 18 (1979), pp. 155-193.

[2] Bollobas B., Fenner T.I. and Frieze A.M., "An Algorithm for Finding Hamiltonain Cycles in a Random Graph," *Proc. 17th Annual ACM Symposium on the Theory of Computing*, (1985) pp. 430-439.

[3] Frieze A.M.,"On Large Matchings and Cycles in Sparse Random Graphs", *Discrete Mathematics*, 59 (1986) pp. 243-256.

[4] Frieze A.M., "On the Exact Solution of Random Travelling Salesman Problems with Medium Size Coefficients,"*SIAM J. of Computing*, 16 (1987) pp. 1052-1072.

[5] Frieze A.M., Personal Communication, Jan, 1992.

[6] Karp R.M., "Probabilistic Analysis of Partitioning Algorithms for the Traveling-Salesman Problem in the Plane," *Math. of Operations Research*, 2 (1977) pp.209-224.

[7] Kern W., "A Probabilistic Analysis of the Switching Algorithm for the Euclidean TSP", *Math. Programming*, 44 (1989) pp.213-219.

[8] Lawler E.L., Lenstra J.K., Rinnoy Kan A.H.G. and Shmoys D.B., eds., *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization*, J. Wiley, New York (1985).

[9] Lin S. and Kernighan B.W., "An Effective Heuristic for the Traveling-Salesman Problem," *Operations Research*, 21 (1973) pp. 498-516.

[10] Lueker G., Manuscript, Stanford University (1984).

[11] Posa L., "Hamiltonian Circuits in Random graphs," *Discrete Math.* 14 (1976) 359-364.

[12] Steele J.M., "Probabilistic Algorithm for the Directed Traveling Salesman Problem," *Math. Oper. Res* 11 (1976) pp. 343-350.

[13] Whitaker L.M., *The Bernoulli Salesman*, Dissertation, Georgia Institute of Technology, 1992.

DIVIDING AND CONQUERING THE SQUARE

Donna C. Llewellyn
Craig A. Tovey
School of Industrial and Systems Engineering
Georgia Institute of Technology
Atlanta, GA 30332-0205

March 10, 1990
Revised April 30, 1991

Abstract

A local minimum of a matrix is a cell whose value is smaller than those of its four adjacent cells. For an $n \times n$ square matrix, we find a local minimum with at most $2.554n$ queries, and prove a lower bound of $\sqrt{2}n$ queries required by any method. For a different neighborhood corresponding to the eight possible moves of a chess king, we prove upper and lower bounds of $3n + O(\log n)$ and $2n$, respectively.

1	2	3	4	5	6	7	8
M	M	M	M	M	M	M	9
27	28	29	30	31	32	M	10
26	M	M	M	M	33	M	11
25	M	39	M	M	34	M	12
24	M	38	37	36	35	M	13
23	M	M	M	M	M	M	14
22	21	20	19	18	17	16	15

Figure 1: Local improvement can require $\sim n^2$ work

1 Introduction

A local minimum of a matrix is a cell with value less than or equal to those of its neighboring cells. How hard is it to find a local minimum of an $n \times n$ matrix? It takes zero computation to determine that one exists, since any global minimum is surely one. And any local minimum, once found, can be verified in $O(1)$ time. On the other hand, local improvement, the most natural search method, can require time $\Omega(n^2)$, the same order as enumeration. For example, suppose a local minimum is sought for a matrix with a descending spiral, illustrated for $n = 8$ in Figure 1 (M is some large value such as n^2). Any local improvement algorithm must traverse a path along that spiral out to the unique local optimum in the corner. But the length of this path will be $\Omega(n^2)$ for most starting points in the square.

The large gap between the obvious $\Omega(1)$ lower bound and $O(n^2)$ upper bound is characteristic of the local optimization problem. In this paper we narrow the gap between these bounds considerably. There are two natural neighborhoods to consider: (i) in the *King adjacency*, the 8 neighbors of a paricular cell are those a king could move to from that cell on a chessboard; (ii) in the *Grid adjacency* the 4 neighbors of a cell are the two adjacent in the same row and the two adjacent in the same column. We will investigate both neighborhoods here. Our best strategies for the two adjacencies turn out rather differently.

First we summarize our principal results: let $\tau(n)$ equal the minimum number of matrix lookups required by any valid algorithm that finds a local optimum of a square $n \times n$ matrix. Then

$$2n \leq \tau(n) \leq 3n + O(\log n) \quad \text{(King Adjacency)}$$

$$\sqrt{2}n \leq \tau(n) \leq 2.554n \qquad \text{(Grid Adjacency)}$$

Our best strategy for the grid adjacency, which yields the $2.554n$ upper bound, is fairly complicated. It is interesting that the matrix, perhaps the simplest natural discrete structure for local optimization, is not very straightforward to solve.

In the rest of this section we review necessary background on search procedures for local optima, and apply it to our specific case of a matrix. The next sections develop the results

stated for the King and Grid adjacencies, respectively. We conclude in Section 4 with some remarks and conjectures.

1.1 Divide-And-Conquer

We seek a strict local optimum of an $n \times n$ matrix A of distinct numbers $A(i,j)$. (All results apply to the slightly more general problem of seeking a nonstrict local optimum when the $A(i,j)$ values are not necessarily distinct.) Equivalently, form a graph $G = (V, E)$ of the matrix as follows: take the cells of A as the nodes V of G, and take the edge set

$$E = [\{(i,j),(i',j')\} : \max(|i-i'|, |j-j'|) = 1]$$

for the king adjacency; and

$$E = [\{(i,j),(i',j')\} : |i-i'| + |j-j'| = 1]$$

for the grid adjacency. Then we seek a local optimum of the function A on the graph G. As in [4],[2],[5], our computational model employs an oracle to compute the values of A. A call to the oracle is a *query*; the total number of queries is taken as the computational effort.

We now summarize necessary background regarding local optima on graphs, from [4]. Results are in terms of finding a local minimum, without loss of generality.

The following *divide-and-conquer* method will find a local minimum of A:

1. Query the vertices in a separating set S of G, finding a vertex $v \in S$ with minimum $A(v)$. (Where a separating set is a collection of vertices which disconnects the graph.)

2. Query the vertices in $N(v)$, *i.e.* those adjacent to v. If v is a local minimum, stop. Otherwise proceed to Step 3.

3. Select $w \in N(v)$ with $A(w) < A(v)$; replace G by the connected component of $G \setminus S$ containing w; return to Step 1.

Virtually all the work in the algorithm occurs in Step 1, where the vertices of the separating sets S are queried. The best separators are found by solving the following

Separation Game

Input: Graph $G = (V, E)$.

Two players: Minimizer I, Maximizer II.

Description: Player I removes vertices from G until it is disconnected. Player II selects one of the newly created components to call G, discards the other components, and passes the new G back to I. The game ends when $|V| \leq 1$.

Step 1: $i = 0, V^0 = V; score(G) = 0.$

Step 2: If $|V| \leq 1$ STOP.

Step 3: Player I chooses $S^i \subseteq V^i$ such that $G^i \setminus S^i$ is not connected or is the empty graph; score(G) = score(G) $+|S^i|$.

Step 4: Player II selects G^{i+1}, a connected component of $G^i \setminus S^i$; $i := i + 1$; Go to Step 2.

The *value* of the separation game on G, denoted $v(G)$, is score(G) when each player plays optimally, I to minimize and II to maximize. Let K denote the number of separating sets used by player I, and let $\lambda_{max}(G)$ denote the maximum degree of any vertex in the graph G. The principal result we employ is

Theorem 1.1.1 *Any algorithm to find a local minumum of G requires at least $v(G)$ queries; the Divide-and-Conquer method requires at most $v(G) + K\lambda_{max}(G)$ queries.*

1.2 Implications

The value of K in Theorem 1.1.1 is typically logarithmically small, and for our graph of the matrix the maximum degree $\lambda_{max}(G) = 8$ or 4. Therefore, the implication of Theorem 1.1.1 is to *transform our problem into an analysis of the separation game on G.*

Solving the separation game is unfortunately NP-Complete in general but we can employ the following partial characterization ([4]):

Lemma 1.2.1 *In the separation game S^i can always be taken to be a minimal separating set of G.*

The minimal separating sets, in turn, are partially characterized in the following Lemma by an interesting dual relationship between the two adjacencies:

Lemma 1.2.2 *A minimal separating set for the separation game under the king adjacency must be connected with respect to the grid adjacency; a minimal separating set under the grid adjacency must be connected with respect to the king adjacency.*

Proof: Let S be a minimal separating set of $G = (V, E)$. S separates some set $U \subset V; U \cap S = \phi; U \neq \phi$ from $V - S - U$ in the graph G. The set U may be taken to be connected for if not we can replace it with any connected component. Here "connected" means under the adjacency for which a local optimum is sought.

Now S must contain $B(U)$, the *boundary* of U, *i.e.* $S \supseteq B(U) \equiv \{v \in V : v \notin U, \exists u \in U, (v, u) \in E\}$ for otherwise there would be a path from U to some vertex in $V - S - U$ that did not pass through S. But also $B(U)$ is a separating set, thus $S = B(U)$ by the minimality of S.

We also may take U to be topologically simple. If U is not simple, there are two cases: (i) if G contains a vertex not encircled by U and not in $B(U)$, then let U' be U together with all vertices encircled by U (thus including some members of $B(U)$). In this case, $B(U')$ is

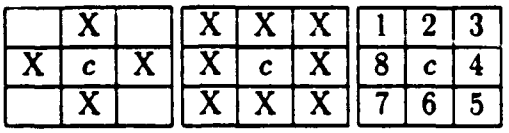

Figure 2: Cell c and its king and grid boundaries

strictly contained in $B(U)$ and therefore $S = B(U)$ was not minimal. Otherwise, (ii) there must exist a nonempty connected component U' of G, encircled by U. Then U' is simple by induction, therefore $B(U') \subseteq B(U)$, and so we can replace U by U'.

It remains to show that when U is connected under the king (respectively grid) adjacency, then $B(U)$ is connected with respect to the grid (respectively king) adjacency. The idea is demonstrated in Figure 2.

The boundary of a cell under the king adjacency is connected with respect to the grid adjacency; the boundary of a cell under the grid adjacency is connected with respect to the king adjacency. For a formal proof, we employ this observation in an induction on $|U|$. Remove c, the rightmost of the uppermost cells of U. Referring to Figure 2, cell $i \notin U : i = 1, \ldots, 4$. By induction, the boundary of $U - c$ is connected as claimed. Again referring to Figure 2, $(5,6,7,8) \cap U \neq \phi$ (king adjacency); $(6,8) \cap U \neq \phi$ (grid). When c is added to $U - c$, the boundary gains all neighbors of c not in U, and loses c since U is connected. Checking all the possible cases, it is generally easy to see that if $B(U - c)$ is connected as claimed, so is $B(U)$. The only non-trivial cases occur when the removal of c disconnects the boundary. For example (king adjacency), if $6 \in U, 7 \notin U, 8 \in U$, then $5 \in B(U), 7 \in B(U)$, and it is possible that 5 and 7 are only connected through c. But then U is not simple, and we have a contradiction. □

Before proceeding with the analysis, it is interesting to see what upper and lower bounds can be derived directly from known results. In [4] it is shown (Corollary 4.11) that a local optimum for any planar graph may be found in $13.35\sqrt{n} + \lambda(\frac{\log n}{\log 3 - 1})$ queries. Since $\lambda = 8$ or 4 here, the logarithmic term is negligible, and we can take $13.35\sqrt{n}$ as an upper bound on the necessary number of queries, for both the king and grid adjacencies. For a lower bound, we appeal to the following results (Corollary 4.5 in [4] and Theorem 11 in [3], respectively):

Theorem 1.2.1 *For any graph and integer t,*

$$v(G) \geq \min\{t, \max_k \min\{|B(S)| : k - t \leq |S| \leq k\}\}.$$

Theorem 1.2.2 *Let $G = (V, E)$ be an $n \times n$ grid graph, and let $A \subset V$ satisfy $|V|/3 \leq |A| \leq 2|V|/3$. Then $|B(A)| \geq n/3$.*

Theorem 1.2.2 applies to the king adjacency, as well, because all edges in the grid graph are edges in the king graph. Letting $t = n/3$ and considering $k = |V|/2 = n^2/2$, we find that $n/3$ is a lower bound on the number of queries needed to find either a king or grid local optimum. Thus we have

Theorem 1.2.3 *Let $\tau(n)$ denote the least number of queries required by a valid algorithm to find a local optimum in a matrix (king or grid adjacency). Then*

$$n/3 \leq \tau(n) \leq 13.35n.$$

We sharpen these bounds considerably in the following.

2 The King Adjacency

We consider the problem of finding a local optimum of a matrix where the neighborhood structure is defined by the king adjacency. By Lemma 1.2.2, a minimal separating set here must be connected with respect to the grid adjacency. We call any such set a region of the matrix. Whenever we consider a region of a matrix, we will think of it as being embedded within a sufficiently large square matrix. For any element in this embedding structure that is not in the original region, define its distance to the region as the length of the shortest path (using grid adjacency) to the region. Then give each of these embedding entries value equal to its distance + n^2. The easiest way to think of this is to think of dropping the region into the embedding structure and hence the values of the surrounding region will be strictly larger than the values within the region and will gradually climb as one moves further away from the region. This will prove useful later when we approximate the indices of a matrix to be queried and may by chance query an entry of a region which does not exist.

2.1 Upper Bounds

Our divide-and-conquer algorithms will have two major types of steps: query and check. For ease of presentation we first define these steps and give the parameters for each. Then we present each procedure, first in words, and then using these generic steps. We also give a pictorial view of each procedure.

A query step takes as input a description of a region of A and gives as output the minimum entry in that region. This step requries a number of queries equal to the size of the input set. This input will be given in one of two ways:

column set (called a Column Query): a pair made up of a column index, j, and a pair of row indices, (i_0, i_1) with $i_0 < i_1$. Here the query step should be performed over rows $i_0, i_0 + 1, \ldots, i_1$ in column j. (We will use the notation Column Query (j, (i_0, i_1): a) where the output of the query is a.)

row set (called a Row Query): a pair made up of a row index, i, and a pair of column indices, (j_0, j_1) with $j_0 < j_1$. Here the query step should be performed over columns $j_0, j_0 + 1, \ldots, j_1$ in row i.

A check step takes as input an entry in the matrix A and gives as output the smallest element among the input and its neighbors.

Our procedure will take as input matrix A and a range of rows and a range of columns, and give as output a local minimum of A within the given ranges.

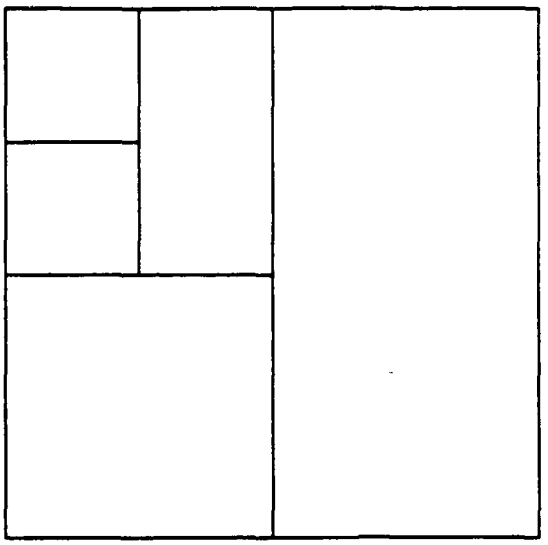

Figure 3: Illustration of Procedure Row-Column

PROCEDURE ROW-COLUMN (Rows$(1,n)$ Columns$(1,n)$: a^*)

This algorithm is the divide-and-conquer algorithm defined in Section 1 with a specific, natural choice of separators. The first separator is the central column of the matrix. If the minimum of this column is not a local optimum then it is assumed without loss of generality that the left neighbor is smaller, and the next separator is the central row of the left submatrix of A. If more separators are needed, the procedure is repeated on the remaining square submatrix (taken without loss of generality to be the upper left submatrix of A.)

Step 1: Column Query $(\lceil \frac{n}{2} \rceil, (1,n) : a^1)$

Step 2: Check $(a^1 : \bar{a}^1)$

Step 3: If $a^1 = \bar{a}^1$ then STOP: $a^* = a^1$. Otherwise, without loss of generality $a^1 = a_{i,j}$ and $\bar{a}^1 = a_{i,j-1}$.

Step 4: Row Query $(\lceil \frac{n}{2} \rceil, (1, \lceil \frac{n}{2} \rceil - 1) : a^2)$

Step 5: Check $(a^2 : \bar{a}^2)$

Step 6: If $a^2 = \bar{a}^2$ then STOP: $a^* = a^2$. Otherwise, without loss of generality $a^2 = a_{i,j}$ and $\bar{a}^2 = a_{i-1,j}$.

Step 7: Procedure Row-Column (Rows$(1, \lceil \frac{n}{2} \rceil - 1)$ Columns$(1, \lceil \frac{n}{2} \rceil - 1)$: a^*)

Theorem 2.1.1 *Procedure Row-Column finds a local minimum of an $n \times n$ matrix in less than $3n + O(\log n)$ queries.*

Proof: Let $f(n)$ be the number of queries that this procedure requires for an $n \times n$ matrix. Then clearly, $f(n) = n + \lceil \frac{n}{2} \rceil + 12 + f(\lceil \frac{n}{2} \rceil - 1)$. This leads to the solution $f(n) = n + 2(\frac{n}{2} + \frac{n}{4} + \ldots) + O(\log n)$ which converges to $3n + O(\log n)$. □

Corollary 2.1.2 *A local optimum of an $m \times n$ matrix, with $m < n$, can be found in less than $m(2 + \alpha) + n/(2^\alpha) + O(\log n) \leq 2m + n + O(\log n)$ queries, where $\alpha \equiv \lfloor \log_2 n/m \rfloor$.*

Proof: Slightly altering Procedure Row-Column to always bisect the longer direction (and hence use the lesser number of queries) in place of alternating between column and row queries gives this result immediately. □

2.2 Lower Bounds

We now find lower bounds for the number of queries needed to find a local minimum of an $n \times n$ matrix. The main result of this section is a lower bound of $2n$ queries.

In the discussions that follow it will be helpful to have the notion of the top, bottom, left and right of a region of a matrix. Suppose that R is a region within $n \times n$ matrix A.

Define

$$\begin{aligned} a &= \min\{i | (i,j) \in R \text{ for any } j, 1 \leq j \leq n\} \\ b &= \min\{j | (a,j) \in R\} \\ c &= \max\{j | (a,j) \in R\} \\ d &= \max\{i | (i,j) \in R \text{ for any } j, 1 \leq j \leq n\} \\ e &= \min\{j | (d,e) \in R\} \\ f &= \max\{j | (d,f) \in R\} \end{aligned}$$

Then define the following "corners" of region R:

$$\begin{aligned} UL \text{ (“upper left”)} &\equiv (a,b) \\ UR \text{ (“upper right”)} &\equiv (a,c) \\ LL \text{ (“lower left”)} &\equiv (d,e) \\ LR \text{ (“lower right”)} &\equiv (d,f) \end{aligned}$$

Now, in order to define the sides, consider the region R and define an entry of R to be interior if it has four grid neighbors within R and frontier otherwise. We will think of traveling along the frontier entries from one corner to another in the clockwise direction (using the grid adjacency to define this path). The collection of frontier entries that one encounters while traveling from UL to UR, inclusive, is called the top, those met while traveling from UR to LR, inclusive, is call the right, those hit while in transit from LR to LL, inclusive, is the bottom, and finally the others, that set lying on the path from LL to UL, inclusive, is the left. It should be clear that if R is the whole matrix A then the top is row 1, the right is column n, the bottom is row n, and the left is column 1. It will not hurt our arguments to have an entry in more than one side.

Now, let the minimum diameter of R, $MinD(R)$, be the length of a simple grid-connected left-right path (i.e. a path of matrix entries from any element of the left to any element of the right) of minimum length in R, where the length of a path is measured by the number

of entries in the path. Analogously define the maximum diameter, $MaxD(R)$. Then, let the minimum height, $MinH(R)$ be the length of a simple grid-connected top-bottom path of minimum length in R; and analogously define the maximum height, $MaxH(R)$.

We will define a strategy for player II, the maximizer in the Separation Game, to get a lower bound on $v(G)$.

PLAYER II STRATEGY

Before Player I plays, we define R to be the subset of matrix A consisting of all unqueried entries. By definition of separation here, this forms a region. Therefore, after Player I's turn, the unqueried entries form two (disconnected) regions, say R_1 and R_2. If one of these R_i, $i = 1$ or 2 touches all four sides (top, bottom, left and right) of R, then choose that component, R_i.

Otherwise, choose that component among R_1 and R_2 which has the largest maximum of its maximum diameter and maximum height. That is, let $d_i = \max\{MaxD(R_i), MaxH(R_i)\}$, and let $i^* = argmax\{d_i\}$. Then choose component R_{i^*}.

Lemma 2.2.1 *It requires at least n queries to find a local minimum of an $n \times n$ matrix.*

Proof: Let the sequence of regions chosen by Player II be given by $A = R^0, R^1, \ldots, R^k$. Then, using the strategy above, for some j, $1 \leq j \leq k$, region R^j will not touch all four sides of region R^{j-1} (since at the end this is true). Hence, either the top and bottom of R^{j-1} are disconnected or the left and right of R^{j-1} are disconnected by queried elements (or both). Without loss of generality, assume that the left and right are disconnected. Then, by piecing together the earlier queries, the left and right of the original matrix, A, are also disconnected. Hence, there exists a path from the top to the bottom of A made up of queried entries. Clearly, these entries alone have used n queries. □

Theorem 2.2.1 *It requires at least $2n - 1$ queries to find a local minimum of an $n \times n$ matrix.*

Proof: First note that the theorem is true in the case of $n = 1$. Consider the above strategy for Player II. Suppose, without loss of generality (as guaranteed by Lemma 2.2.1), that eventually Player I queries a top-bottom path. Consider the first time such a path has been queried (i.e., this is the first time Player II must choose a component that does not touch all four sides). Suppose that up to this time $n + H$ queries have been made. If $H > n$, then clearly the theorem is proved. So suppose that $H < n$. Now, in the chosen component, there exists a square of side length equal to the minimum of the minimum diameter and minimum height of the component. How can this value be made as small as possible? By using all of the extra H queries to shorten one of them, say the minimum diameter. This is done by using all of these queries in "horizontal" queries, and centering them so that the minimum diameter is exactly $\frac{n-H}{2}$. Hence, in the chosen component, there exists a square matrix of size at least $\frac{n-H}{2} \times \frac{n-H}{2}$. By Lemma 2.2.1 above, this requires at least $\frac{n-H}{2}$ queries. Thus, the total number of queries so far is at least $n + H + \frac{n-H}{2} = 1.5n + .5H$. Now, "bootstrapping" with this result in place of the bound given in the lemma gives that the enclosed square

requires at least $1.5\left(\frac{n-H}{2}\right)$ and hence the total lower bound is $1.75n + .25H$. Iterating this procedure gives a lower bound of $2n$ queries. □

Now we give an alternative way to arrive at the same lower bound of $2n$ queries. We include this because the method is quite different and we believe it provides some additional insight into the geometry of the problem.

First we need the following lemma. In this method, it is best to think of the matrix, A, as being placed in the $\mathcal{R}^2$ plane in the following way. Each entry takes up a unit square, so that the outer edge of the left of A is the y-axis, the outer edge of the bottom of A is the x-axis, the outer edge of the right of A is the line $x = n$ and the outer edge of the top of A is the line $y = n$. Then, given any region R, its area $Area(R)$ is the actual (continous) area of the enclosed region; its perimeter $Per(R)$ is the sum of the euclidean lengths of all the straight lines that make up the outer edges of the frontier of the region.

Lemma 2.2.2 *For any region, R,*

$$\frac{\sqrt{Area(R)}}{Per(R)} \leq \frac{1}{4}$$

Proof: First consider a rectangle with width w and length $w + \delta$, where $\delta \geq 0$. Then $Per(R) = 2w + 2(w + \delta)$ and hence $(Per(R))^2 = 16w^2 + 4\delta^2 + 16w\delta$. Further, $Area(R) = w(w + \delta)$, so $16(Area(R)) = 16w^2 + 16w\delta$. Since $\delta \geq 0$, it is clear then that $(Per(R))^2 \geq 16Area(R)$ so the lemma follows for rectangles.

Now consider any region R. Let the tightest circumscribing rectangle be T. It is clear that $Area(T) \geq Area(R)$. Hence it is sufficient to prove that $Per(T) \leq Per(R)$ since then we will have $(Per(R))^2 \geq (Per(T))^2 \geq 16Area(T) \geq 16Area(R)$. Think of traveling around the boundary of R and notice that this boundary will agree with the boundary of T at least for some stretch on each side of T (top, bottom, left and right). Where the boundary of R differs from the boundary of T, call it a journey. Any journey either originates and ends on the same side or else it originates on one side and ends on an adjacent side of T. We will consider each of these cases separately.

1. Suppose the journey originates and terminates on the same side. Without loss of generality, suppose it is the top or bottom. Consider the origin as a point in the $\mathcal{R}^2$ plane, (a, b). Then the terminus is another point (c, b). Without loss of generality assume that $c > a$. Then the amount of the perimeter of T between these two points is exactly $c - a$. The amount of the perimeter of R that lies between these points is at least $c - a$ since the boundary follows the grid adjacency (it might also have a vertical component and hence could be greater).

2. Now without loss of generality, suppose that the journey originates on the left and ends on the top. Then the origin is some point (a, b) and the terminus is another point (c, d). We know that $d > b$. Then the perimeter of T between these points is $(d - b) + |(c - a)|$ and the perimeter of R must be at least this by the same reasoning as above (it must travel at least $d - b$ vertical distance and at least $|c - a|$ horizontal distance).

Since the two regions clearly have the same perimeter between journeys, this completes the proof. □

Now consider the region R before Player I queries during some iteration of the separation game. The queries that follow before another turn of Player II can be combined into some separating set, C. This causes R to be divided up into two regions, R_1 and R_2. For ease of presentation, call $Area(R) \equiv A$, $Per(R) \equiv P$ and similarly, $Area(R_i) \equiv A_i$ and $Per(R_i) \equiv P_i$ for $i = 1, 2$. We will abuse notation slightly and use C also to represent the number of entries in the set C. We now need the following result.

Lemma 2.2.3 *Let*

$$\frac{A_1}{P_1} = \max_{i=1,2} \left\{ \frac{A_i}{P_i} \right\},$$

and suppose that

$$A_1 = \lambda A \text{ for some } 0 < \lambda < 1.$$

Then,

$$P_1 - \lambda P \leq 2\lambda C.$$

Proof: First we will need the following inequality: $P_1 + P_2 \leq 2C + P$. To see this rigorously, we need the following notation. Let the frontier of region R_i for $i = 1, 2$ be denoted F_i. Let F_i be the (not necessarily disjoint) union of F_{i1} and F_{i2}, where F_{i1} is the part of F_i which is adjacent to the frontier of C, and F_{i2} is the part of F_i which is a subset of the frontier of R. We can break up $P_1 + P_2$ into the part which arises from tracing along $F_{12} \cup F_{22}$ and the part which comes from tracing along $F_{11} \cup F_{21}$. The first part is clearly less than or equal to P. Our goal is therefore to show that the second part is at most $2C$. To this end, C can be assumed to be a minimal separating set by Lemma 1.2.1, whence simple case analysis verifies that no cell of C has more than 2 sides adjacent to $F_{11} \cup F_{21}$. Lemma 1.2.1 also ensures that C has no interior. Therefore, the second part is at most $2C$ and the inequality holds.

Thus,

$$2C + P \geq P_1 + P_2$$

and so

$$P_1 + P_2 - 2C \leq P.$$

This implies that

$$\begin{aligned} P_1 - \lambda P &\leq P_1 - \lambda(P_1 + P_2 - 2C) \\ &\leq (1 - \lambda)P_1 - \lambda P_2 + 2\lambda C. \end{aligned}$$

So, if we can show that

$$\lambda P_2 \geq (1 - \lambda)P_1$$

then the lemma is proved.

By assumption,

$$\begin{array}{rcl} \frac{A_1}{P_1} & \geq & \frac{A_2}{P_2} \\ \Rightarrow \quad \frac{\lambda A}{P_1} & \geq & \frac{(1-\lambda)A}{P_2} \\ \Rightarrow \quad \lambda P_2 A & \geq & (1-\lambda)AP_1 \\ \Rightarrow \quad \lambda P_2 & \geq & (1-\lambda)P_1. \end{array}$$

□

Now we are ready for our main result.

Theorem 2.2.2 *The number of queries needed to find a local optimum in a region of area A and perimeter P is at least*

$$\frac{8A}{P} - 1$$

Proof: The proof is inductive on A. First note that if $A = 1$ then it must be that $P = 4$, and clearly it requires exactly one query to solve the problem, so the result holds.

In general, it is sufficient to show that

$$\frac{8A}{P} \leq C + \frac{8A_1}{P_1}$$

where as in Lemma 2.2.3, $\max_{i=1,2}\left\{\frac{A_i}{P_i}\right\} = \frac{A_1}{P_1}$. By Lemma 2.2.2 we know that

$$P \geq 4\sqrt{A}$$

and that

$$P_1 \geq 4\sqrt{A_1} = 4\sqrt{A\lambda}.$$

So, $P \times P_1 \geq 16A\sqrt{\lambda}$. This implies that $C\sqrt{\lambda}(P \times P_1) \geq 16AC\lambda$. Rearranging this gives

$$\frac{C\sqrt{\lambda}}{8A} \geq \frac{2C\lambda}{P \times P_1}.$$

Using Lemma 2.2.3 gives the righthand side is

$$\geq \frac{P_1 - \lambda P}{P \times P_1} = \frac{1}{P} - \frac{\lambda}{P_1}.$$

So,

$$\frac{C\sqrt{\lambda}}{8} \geq \frac{A}{P} - \frac{A\lambda}{P_1} = \frac{A}{P} - \frac{A_1}{P_1}.$$

Hence

$$\frac{8A}{P} \leq C\sqrt{\lambda} + \frac{8A_1}{P_1}.$$

We know that $\lambda < 1$ so the result follows. □

Corollary 2.2.3 *The number of queries needed to find a local minimum of an $n \times n$ matrix is at least $2n$.*

Proof: For an $n \times n$ square $A = n^2$ and $P = 4n$. Using this in Theorem 2.2.2 above gives the result immediately. □

3 The Grid Adjacency

In this section we consider the problem of finding a local optimum of a matrix where the neighborhood structure is defined by the usual adjacency in grids. By Lemma 1.2.2, a minimal separating set need only be connected with respect to the king adjacency. Thus in this section a region of a matrix will be a subset of entries so connected. As in Section 2 we will continue to think of our matrix as being embedded within a larger square matrix. Of course, we must update our definition of distance to be consistent with king adjacency connected paths here.

3.1 Upper Bounds

Of course, Procedure Row-Column can still be used here since any set of entries that is connected with respect to grid adjacency will also be connected with respect to king adjacency. Hence we immediately get an upper bound of $3n + O(\log n)$ on the number of queries needed. Here, though, we can show that it is not optimal. To improve on it, we employ diagonal queries. The intuition here is that while the euclidean length of a diagonal of an $n \times n$ matrix is $n\sqrt{2}$, there are only n entries in the matrix on the diagonal, so diagonal queries are more efficient by a factor of $\sqrt{2}$.

In this section we will need one further way to call a query step:

line segment (called a Line Query): a pair made up a line definition, $bx + cy = d$, and a range on x, $x_0 \leq x \leq x_1$. Here it should be interpreted that the matrix, A, is placed in the $\mathcal{R}^2$ plane with the $(n, 1)$ entry at the origin and the $(1, n)$ entry at the $(1, 1)$ position in the plane. The query should be performed at the intersections of the matrix and the line segment.

Notice that the Row and Column Queries can be described as special cases of the Line Query. However, for technical reasons we leave them with their own descriptions.

To make the following procedures easier to understand, we must first take care of rotated matrices. We will call a square matrix that has been rotated 45 degrees a diamond matrix. Let $A = [a_{ij}]$ be our $n \times n$ matrix and let $B = [b_{ij}]$ be the inscribed diamond matrix. Note that B has euclidean side lengths equal to $n/\sqrt{2}$, but when counting queries the side length is effectively only $n/2$. For this reason we will refer to B as an $\frac{n}{2} \times \frac{n}{2}$ diamond matrix inscribed in the $n \times n$ matrix A. For ease, let n be even. Take $b_{11} = a_{\frac{n}{2},1}$, $b_{1,\frac{n}{2}} = a_{1,\frac{n}{2}}$, $b_{\frac{n}{2},1} = a_{n,\frac{n}{2}}$, and $b_{\frac{n}{2},\frac{n}{2}} = a_{\frac{n}{2},n}$. Then to find a local minimum of B, we will perform Procedure Row-Column on it, but querying the appropiate diagonal segments of A in place of rows and columns. The input will be the matrix A, but it is understood that only the elements of B must be known, and the check queries should only be performed over elements of B.

PROCEDURE DIAMOND (Rows$(1, n)$ Columns$(1, n)$: a^*)

Step 1: Line Query $(x + y = 1, \frac{1}{4} \leq x \leq \frac{3}{4} : a^1)$

Step 2: Check $(a^1 : \bar{a}^1)$

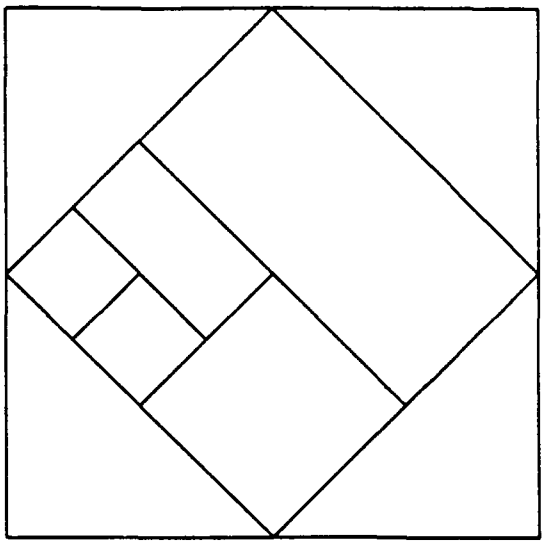

Figure 4: Procedure Diamond

Step 3: If $a^1 = \bar{a}^1$ then STOP: $a^* = a^1$. Otherwise, without loss of generality $a^1 = a_{i,j}$ and $\bar{a}^1 = a_{i,j-1}$ or $a_{i+1,j}$.

Step 4: Line Query $(x = y, \frac{1}{4} \leq x \leq \frac{1}{2} : a^2)$

Step 5: Check $(a^2 : \bar{a}^2)$

Step 6: If $a^2 = \bar{a}^2$ then STOP: $a^* = a^2$. Otherwise, without loss of generality $a^2 = a_{i,j}$ and $\bar{a}^2 = a_{i+1,j}$ or $a_{i,j+1}$.

Step 7: Procedure Diamond(Rows$(\lceil \frac{n}{2} \rceil + 1, n)$ Columns$(\lceil \frac{n}{4} \rceil, \lceil \frac{3n}{4} \rceil : a^*)$

Corollary 3.1.1 *Procedure Diamond finds a local minimum of an* $\frac{n}{2} \times \frac{n}{2}$ *diamond matrix in less than* $1.5n + O(\log n)$ *queries.*

Proof: This follows immediately from Theorem 2.1.1 and the discussion above on diamond matrices. □

Corollary 3.1.2 *A local minimum of an* $m \times n$ *diamond matrix, with* $m \leq n$, *can be found in less than* $m(2+\alpha+n/(2^\alpha m))+O(\log n) \leq 2m+n+O(\log n)$ *queries, where* $\alpha \equiv \lfloor \log_2 n/m \rfloor$.

□

To make the presentation easier, when we call Procedure Diamond we will refer to this more general form of Corollary 3.1.2 which can take as input an oblong diamond matrix. We will call the procedure by giving as input the four corners of the matrix rather than the rows and columns of the embedding square (or rectangular) matrix.

One last result we need before we can use this as a subroutine is the principle of containment: If one region, B, is a subset of another region, A, then it can require no more

queries to find a local minimum of B than to find one of A. This is clear, since to find a local optimum of B we could just consider B to be embedded within A as discussed above (of course A in turn is embedded within another large matrix), and then find a local optimum of A. We will sometimes call a procedure on a row and column set that imply that the whole diamond matrix does not exist (it would require rows or columns with negative indices or indices greater than n). When we do this we are actually relying on this containment principle, and are considering the existing region to be embedded within the called diamond matrix.

PROCEDURE DIAGONAL (Rows$(1,n)$ Columns$(1,n)$: a^*)

This algorithm first queries and checks along the NE-SW diagonal of the square matrix. Failing to find a local optimum here, it queries and checks along half of the NW-SE diagonal. Then if it still hasn't found a local optimum it queries a diagonal paralled to the NE-SW line halfway down the triangle. After this it is either left with another triangle which it treats with Procedure Diamond, or it forms a diamond and a triangle out of the resulting shape. Each of these can be taken care of with Procedure Diamond.

Step 1: Line Query $(x = y, 0 \leq x \leq 1 : a^1)$

Step 2: Check $(a^1 : \bar{a}^1)$

Step 3: If $a^1 = \bar{a}^1$ then STOP: $a^* = a^1$. Otherwise, without loss of generality $a^1 = a_{i,j}$ and $\bar{a}^1 = a_{i-1,j}$ or $a_{i,j-1}$.

Step 4: Line Query $(x + y = 1, 0 \leq x \leq \frac{1}{2} : a^2)$

Step 5: Check $(a^2 : \bar{a}^2)$

Step 6: If $a^2 = \bar{a}^2$ then STOP: $a^* = a^2$. Otherwise, without loss of generality $a^2 = a_{i,j}$ and $\bar{a}^2 = a_{i,j-1}$ or $a_{i,j+1}$.

Step 7: Line Query $(y = x + \frac{1}{2}, 0 \leq x \leq \frac{1}{4} : a^3)$

Step 8: Check $(a^3 : \bar{a}^3)$

Step 9: If $a^3 = \bar{a}^3$ then STOP: $a^* = a^3$. Otherwise,

Case 1: $a^3 = a_{i,j}$ and $\bar{a}^3 = a_{i,j-1}$ or $a_{i-1,j}$, then Procedure Diamond $((0,1),(0,\frac{1}{2}),(\frac{1}{4},\frac{3}{4}): a^*)$

OR

Case 2: $a^3 = a_{i,j}$ and $\bar{a}^3 = a_{i,j+1}$ or $a_{i+1,j}$, then go to Step 10.

Step 10: Line Query $(y = \frac{1}{2} - x, 0 \leq x \leq \frac{1}{4} : a^4)$

Step 11: Check $(a^4 : \bar{a}^4)$

Step 12: If $a^4 = \bar{a}^4$ then STOP: $a^* = a^4$. Otherwise,

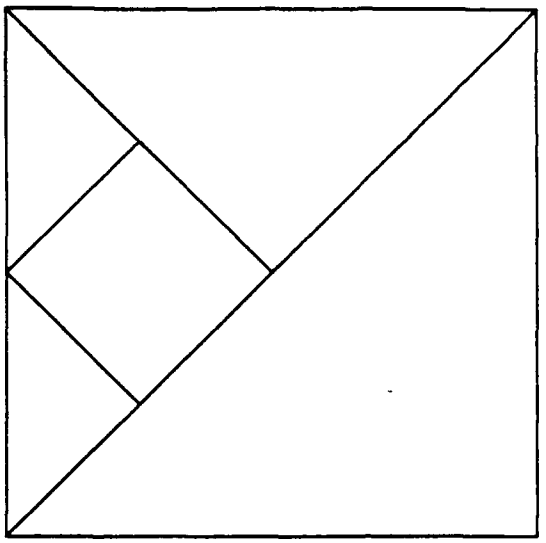

Figure 5: Procedure Diagonal

Case 1: $a^4 = a_{i,j}$ and $\bar{a}^4 = a_{i,j-1}$ or $a_{i+1,j}$, then Procedure Diamond $((0,\frac{1}{2}), (\frac{1}{4},\frac{1}{4}), (0,0)$: $a^*)$

OR

Case 2: $a^4 = a_{i,j}$ and $\bar{a}^4 = a_{i,j+1}$ or $a_{i-1,j}$, then Procedure Diamond $((0,\frac{1}{2}), (\frac{1}{4},\frac{3}{4}), (\frac{1}{4},\frac{1}{4}), (\frac{1}{2},\frac{1}{2})$: $a^*)$

Theorem 3.1.3 *Procedure Diagonal finds a local optimum of an $n \times n$ matrix in less than $2.75n + O(\log n)$ queries.*

Proof: This procedure terminates in a Procedure Diamond iteration in either Case 1 of Step 9 or Case 1 or 2 of Step 12. We will consider each of these in turn. The Procedure Diamond iteration of each of these steps has as input a region within an $\frac{n}{4} \times \frac{n}{4}$ diamond matrix (within an $\frac{n}{2} \times \frac{n}{2}$ square matrix) and hence by Corollary 3.1.1 requires no more than $\frac{3n}{4} + O(\log n)$ queries. The number of queries up to Step 9 is $n + \frac{n}{2} + \frac{n}{4} + 12$. Hence if the procedure terminates here then the total number of queries is less than $2.5n + O(\log n)$. If, however, the procedure reaches Step 12 then it already has performed $n + \frac{n}{2} + \frac{n}{4} + \frac{n}{4} + 16$ queries and hence the total number is less than $2.75n + O(\log n)$ if the termination occurs in either Case 1 or 2 of Step 12. □

The problem with this algorithm is that it is not "balanced." That is, one sees that if termination occurs in Step 9 then the total number of queries is significantly less than if termination occurs in Step 12. Intuitively, we should balance the regions to be explored, so that the number of queries is approximately the same regardless of where the algorithm is led. The place where we have the leeway to do this in Procedure Diagonal is when we choose the third query (Step 7). It was rather arbitrary that we decided to query the halfway diagonal. With this in mind we now introduce a "generic" or parameterized algorithm that leaves as

parameters where the third (and later) diagonals should be queried. Then we optimize over possible values of these parameters in order to balance the resulting regions and so minimize the total number of queries.

We will need one subroutine that we haven't seen yet, a procedure to deal with triangles that doesn't use Procedure Diamond directly. Instead, this algorithm iterates the ideas within Procedure Diagonal with the parametric argument described above. Its input will be the three corners of the triangle. It is assumed that the triangle is an isosceles right triangle.

PROCEDURE TRIANGLE $((0,1), (0,0), (\frac{1}{2}, \frac{1}{2})$: $a^*)$

This algorithm first queries the diagonal that is α of the way down the triangle (Procedure Diagonal uses $\alpha = \frac{1}{2}$). This diagonal divides the triangle into a triangle and a trapezoid. If the diagonal does not turn up a local optimum then the algorithm will either iterate on the new triangle portion or it needs to deal with the trapezoid. Recall that Procedure Diagonal took care of the trapezoid by dividing it into a triangle and a diamond. Here, it first cuts it into two with a NW-SE diagonal β of the way from the first cut (Procedure Diagonal uses $\beta = \frac{1}{2}$). Now, since β doesn't necessarily equal $\frac{1}{2}$, this diagonal cut divides the trapezoid into a region that can be handled by Procedure Diamond and another trapezoid. This new trapezoid is again divided by a NW-SE diagonal, this time γ of the way down. Finally, this results in two regions, one of which can be taken care of with Procedure Diamond and the other by iterating Procedure Triangle.

Step 1: Line Query $(y = x + (1-\alpha), 0 \leq x < \frac{\alpha}{2} : a^1)$

Step 2: Check $(a^1 : \bar{a}^1)$

Step 3: If $a^1 = \bar{a}^1$ then STOP: $a^* = a^1$. Otherwise,

Case 1: $a^1 = a_{i,j}$ and $\bar{a}^1 = a_{i,j-1}$ or $a_{i-1,j}$, then Procedure Triangle $((1,0), (0,1-\alpha), (\alpha, 1-\alpha)$: $a^*)$

OR

Case 2: $a^1 = a_{i,j}$ and $\bar{a}^1 = a_{i,j+1}$ or $a_{i+1,j}$, then go to Step 4.

Step 4: Line Query $(y = -x + (1-2\beta), -\beta + \frac{\alpha}{2} \leq x < \frac{1}{2} - \beta : a^2)$

Step 5: Check $(a^2 : \bar{a}^2)$

Step 6: If $a^2 = \bar{a}^2$ then STOP: $a^* = a^2$. Otherwise,

Case 1: $a^2 = a_{i,j}$ and $\bar{a}^2 = a_{i,j+1}$ or $a_{i-1,j}$, then Procedure Diamond $((\frac{\alpha}{2} - \beta, 1 - \frac{\alpha}{2} - \beta), (\alpha, 1-\alpha), (\frac{1}{2} - \beta, \frac{1}{2} - \beta), (\frac{1}{2}, \frac{1}{2})$: $a^*)$

OR

Case 2: $a^2 = a_{i,j}$ and $\bar{a}^2 = a_{i,j-1}$ or $a_{i+1,j}$, then go to Step 7.

Step 7: Line Query $(y = -x + (1 - 2\beta - 2\gamma), \min(0, -\frac{1}{4} - \frac{\beta}{2} - \frac{\gamma}{2} + \frac{\alpha}{2}) \leq x < \frac{1}{2} - \beta - \gamma : a^3)$

Step 8: Check $(a^3 : \bar{a}^3)$

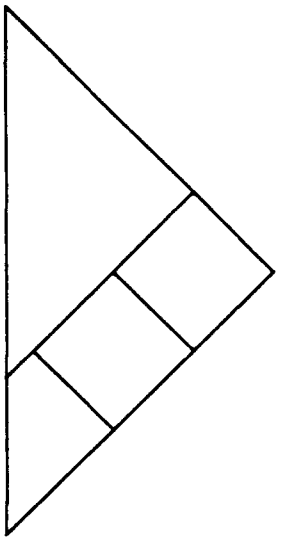

Figure 6: Procedure Triangle

Step 9: If $a^3 = \bar{a}^3$ then STOP: $a^* = a^3$. Otherwise,

Case 1: $a^3 = a_{i,j}$ and $\bar{a}^3 = a_{i,j+1}$ or $a_{i-1,j}$, then Procedure Diamond $((0, 1-\alpha), (\frac{\alpha}{2}-\beta, 1-\frac{\alpha}{2}-\beta), (\frac{1}{2}-\beta-\gamma, \frac{1}{2}-\beta-\gamma), (\frac{1}{2}-\beta, \frac{1}{2}-\beta)$: $a^*)$

OR

Case 2: $a^3 = a_{i,j}$ and $\bar{a}^3 = a_{i,j-1}$ or $a_{i+1,j}$, then Procedure Triangle $((0, 1-\alpha), (0,0), (\frac{1}{2}-\beta-\gamma, \frac{1}{2}-\beta-\gamma)$: $a^*)$

Now we will use this procedure as a subroutine to get a parameterized form of Procedure Diagonal. This procedes exactly like Procedure Diagonal except that once we are left with a triangle to analyze we use Procedure Triangle rather than the unparameterized version (which as mentioned above set α and β equal to $\frac{1}{2}$ and has no γ.)

PROCEDURE PARA-DIAGONAL (Rows$(1,n)$ Columns$(1,n)$: a^*)

As mentioned above, this procedure starts out like Procedure Diagonal, querying the NE-SW diagonal and then the half NW-SE diagonal. Now it is left with an isosceles right triangle and so finishes by calling Procedure Triangle.

Step 1: Line Query $(y = x, 0 \leq x \leq 1 : a^1)$

Step 2: Check $(a^1 : \bar{a}^1)$

Step 3: If $a^1 = \bar{a}^1$ then STOP: $a^* = a^1$. Otherwise, without loss of generality $a^1 = a_{i,j}$ and $\bar{a}^1 = a_{i-1,j}$ or $a_{i,j-1}$.

Step 4: Line Query $(y = -x + 1, 0 \leq x \leq \frac{1}{2} : a^2)$

Step 5: Check $(a^2 : \bar{a}^2)$

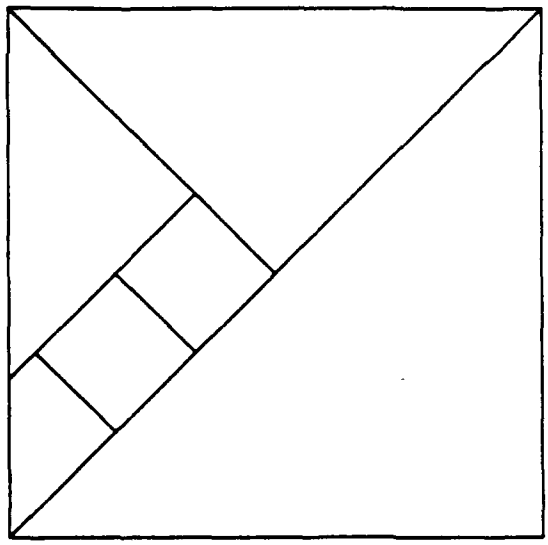

Figure 7: Procedure Para-Diagonal

Step 6: If $a^2 = \bar{a}^2$ then STOP: $a^* = a^2$. Otherwise, without loss of generality $a^2 = a_{i,j}$ and $\bar{a}^2 = a_{i,j-1}$ or $a_{i,j+1}$.

Step 7: Procedure Triangle $((1,0), (0,0), (\frac{1}{2}, \frac{1}{2})$: $a^*)$

Theorem 3.1.4 *Procedure Para-Diagonal finds a local optimum of an $n \times n$ matrix in less than $2.5445n + O(\log n)$ queries.*

Proof: For this procedure to converge we must restrict the values of the various parameters. We will always require that α is between $\frac{1}{4}$ and $\frac{1}{2}$, β is at least $\frac{1}{2} - \alpha$, and γ is no more than $\frac{1}{2} - \alpha$. To analyze this, we must separately consider the four different ways that this algorithm can terminate. These are:

1. If Case 1 is always chosen in Step 3 of Procedure Triangle;
2. At any time Case 1 is chosen in Step 6 of Procedure Triangle;
3. At any time Case 1 is chosen in Step 9 of Procedure Triangle;
4. Case 2 is chosen in Step 9 of Procedure Triangle.

First, it is clear that the largest number of queries will result in any of the three last choices if they occur at the first possible time, i.e., the first time that step of Procedure Triangle is encountered. Indeed, the number of queries will decrease the later they are chosen. Hence, we will analyze the possibilities above, with options (2) - (4) understood to mean that they actually occur at the first time that step is encountered in the first iteration of Procedure Triangle. Since we wish to optimize over choices of parameters α, β, and γ, we will first symbolically write down the number of queries in parametric form and then discuss possible

values of these parameters. Note that the first 6 steps of Procedure Para-Diagonal require $1.5n + 8$ queries. Hence, we will only analyze Procedure Triangle (with the inputs used in Step 7 of our procedure) and at the end will add these extra queries.

1. Case 1 of Step 3 is always chosen (Procedure Triangle is iterated): Here it is straightforward that the total number of queries is less than $.5\left\lceil\frac{\alpha}{\frac{1}{2}-\alpha}\right\rceil n + O(\log n)$.

2. Case 1 of Step 6 is chosen: The procedure Diamond step will require less than $[2(\frac{1}{2} - \alpha) + \beta]n + O(\log n)$ queries by Corollary 3.1.2. The steps of Procedure Triangle leading up to this step require $[\alpha + (\frac{1}{2} - \alpha)]n + 8$ queries. Hence this termination option requires less than $1.5n + [-2\alpha + \beta]n + O(\log n)$ queries.

3. Case 1 of Step 9 is chosen: Here the Procedure Diamond step will require less than $[2\gamma + (\frac{1}{2} - \alpha)]n + O(\log n)$ queries. The steps of Procedure Triangle leading up to this termination option require $[\alpha + (\frac{1}{2} - \alpha) + (\frac{1}{2} - \beta - \gamma)]n + 12$ queries. Hence this termination option requires less than $1.5n + [-\alpha - \beta + \gamma]n + O(\log n)$ queries.

4. Case 2 of Step 9 is chosen: Here the triangle left to analyze requires no more than $\tau[\frac{1}{2} - \beta - \gamma]n$ queries where τkn is the number of queries needed by Procedure Triangle on an isosceles right triangle with side length kn. The steps of Procedure Triangle leading up to this step are as given in (3) above. Hence this termination option requires less than $n - \beta n - \gamma n + \tau[\frac{1}{2} - \beta - \gamma]n + O(\log n)$ queries.

As an example, consider the values of $\alpha = \frac{1}{3}$, $\beta = \frac{1}{4}$ and $\gamma = \frac{1}{6}$. The reader can easily verify that these satisfy our convergence requirements stated above. These values give the following results:

1. $1.0n + O(\log n)$

2. $1.08n + O(\log n)$

3. $1.08n + O(\log n)$

4. Using $\tau n = 1.25n$ from Procedure Diagonal (the number of queries less the first two query and check steps) above, gives $.70n + O(\log n)$

Using these values, we would have an upper bound of $2.58n + O(\log n)$ queries for the whole problem. Notice that we have almost accomplished complete balancing of the regions here. It is probably too much to ask for to also balance the last triangular region. Next, we discuss how to pick good values for α, β and γ.

In order to optimize the parameters α, β, and γ, we will write them each in terms of $.5\tau$. What we will show is that the smallest possible $.5\tau$ we can get using this procedure is between 1.044 and 1.045.

Since we are trying to balance the different regions, what we will do is set each of the results (1) - (3), above equal to $.5n\tau$. Then we will require that the quantity in (4) remains no more than $.5n\tau$. We are suppressing the check steps and their $O(\log n)$ terms for clarity.

$$\begin{array}{lcccc}
(1) & & .5n\left[\frac{\alpha}{\frac{1}{2}-\alpha}\right] & = & .5n\tau \\
 & \Rightarrow & \alpha & = & \left[\frac{.5\tau}{1+2(.5\tau)}\right] \\
 & & & & \\
(2) & & [1.5n - 2\alpha + \beta]n & = & .5n\tau \\
 & \Rightarrow & \beta & = & \left[\frac{2(.5\tau)^2-1.5}{2(.5\tau)+1}\right] \\
 & & & & \\
(3) & & [1.5 - \alpha - \beta + \gamma]n & = & .5n\tau \\
 & \Rightarrow & \gamma & = & \left[\frac{4(.5\tau)^2-.5\tau-3}{2(.5\tau)+1}\right] \\
 & & & & \\
(4) & & 1 - \beta - \gamma - \tau[.5 - \beta - \gamma] & \leq & .5\tau \\
 & \text{and} & .5\tau & \geq & 0
\end{array}$$

This gives a third order equation to solve. The solution is $.5\tau = 1.0445$. Note that we must also make sure that our convergence ranges on the parameters are enforced ($.25 \leq \alpha \leq .5$, $\beta \geq .5 - \alpha$, and $\gamma \leq .5 - \alpha$). Checking these with the above gives the information that $1 \leq .5\tau \leq 1.075$. Hence we are within the necessary bounds. Using this value of $.5\tau = 1.0445$ gives

$$\begin{aligned}
\alpha &\cong .33813532 \\
\beta &\cong .22077064 \\
\gamma &\cong .10340596
\end{aligned}$$

As these satisfy all of our requirements, this is the solution. The total number of queries for the $n \times n$ matrix is less than $[1.5 + 1.0445]n + O(\log n) = 2.\text{xxx}n + O(\log n)$, completing the proof of Theorem 3.1.4. □

3.2 Lower Bounds

Here notice that we can not use the results in Section 2.2 directly since more sophisticated query sets may be chosen by the Player I with this adjacency structure. However, from our discussion on diagonals in Section 3.1, Theorem 2.2.1 does immediately give the following result.

Theorem 3.2.1 *It requires at least $n\sqrt{2}$ queries to find a local minimum of an $n \times n$ matrix.*

Adjacency	Lower Bound	Upper Bound
King	$2n$	$3n$
Grid	$\sqrt{2}n$	$2.5445n$

Figure 8: Queries to find a local optimum in an $n \times n$ square

4 Conclusions

4.1 Conjectures

The bounds found in Sections 2 and 3 are summarized in Figure 8 (logarithmic terms are disregarded here). We have substantially improved on the bounds of Theorem 1.1.1, but a gap persists for both adjacencies. In particular, we conjecture a lower bound of $3n$ for the king adjacency (this would imply $3/\sqrt{2}$ for the grid). Neither proof of Theorem 2.2.1 applies directly to this conjecture. The first proof, at the least, would require a new strategy for Player II. (Against the given strategy, player I can achieve close to $2n$ by separating an $(n-2) \times (n-2)$ non-centered square from the rest of the $n \times n$ square.) To prove the conjecture with the second method, one would show that at least $12A/P$ queries are required to find a local optimum in a region of area A and perimeter P. However, this is false: consider a $1 \times \sqrt{2}$ rectangle. By Corollary 2.1.2, a local minimum can be found in $2+\sqrt{2}$ queries. Now $A = \sqrt{2}$, $P = 2\sqrt{2}+2$; so $(2+2\sqrt{2})P/A = (2+\sqrt{2})^2 = 11.656\ldots < 12$. We do conjecture that at least $8\sqrt{2}A/P$ queries are required to find a local optimum under the king adjacency. We also conjecture a lower bound of $2.5n$ for a square matrix under the grid adjacency.

Since completing an earlier draft of this paper, we have found that Althöfer and Koschnick [1] have independently studied the problem of local optimization on an m-dimensional grid. For $m = 2$, (our grid adjacency case), their results reduce to

$$\frac{n^2}{4n+1} = \frac{n}{4} - \frac{1}{16} + o(n) \leq \tau(n) \leq 4n + O(\log n).$$

It would be interesting to see if Theorems 3.1.4 and 3.2.1 could be extended to $m = 3$ or more dimensions to strengthen the bounds in [1].

4.2 Related problems

Local saddlepoints are related to local optima with the grid adjacency. We say a point (i, j) is a local saddlepoint iff

$$A(i \pm 1, j) < A(i, j) < A(i, j \pm 1),$$

i.e., $A(i, j)$ is larger than its two horizontal grid neighbors and smaller than its two vertical grid neighbors. Unlike local optima, local saddlepoints need not exist. Finding them turns out to be more costly, as the following theorem shows.

Theorem 4.2.1 *Any valid local saddlepoint algorithm requires at least $nm/4$ queries in the worst case for an $n \times m$ matrix.*

	1		1		1
4	5	4	5	4	5
	1		1		1
4	5	4	5	4	5

Figure 9: Adversary's matrix

	saddlepoint	local optimum
row-column	$\theta(n)$	$\theta(n^2)$
grid	$\theta(n^2)$	$\theta(n)$

Figure 10: Comparison between grid and row-column adjacencies

Proof: We play the adversary against an arbitrary valid algorithm. We let A be made of identical 2×2 submatrices as shown in Figure 9 Each blank cell will have value either 3 or 7, but we do not decide which until it is queried. With this strategy, none of the fixed cells can be a local saddlepoint, and each unfixed cell is a local saddlepoint iff its value is 3. Now, as the algorithm makes queries, we respond with the fixed value for fixed cells, and with 7 for the unfixed cells, until the last unfixed cell is queried. Then we randomly decide on either 3 or 7. Obviously the algorithm must query all $nm/4$ unfixed cells to determine whether or not a local saddlepoint exists. □

Corollary 4.2.2 *It requires $\theta(n^2)$ queries to find a local saddlepoint.*

Proof: Obviously there exists a valid $O(n^2)$ algorithm, and the result follows. □

In a broader context, Theorem 4.2.1 displays a nice asymmetry between grid and row-column adjacencies. In the row-column adjacency, a cell is adjacent to all other cells in its row or column. That is, the graph has edge set

$$\{(i,j),(i',j')\} \in E \Leftrightarrow \min\{|i-i'|,|j-j'|\} = 0.$$

A (row-column) saddlepoint is the largest in its row and smallest in its column. Bounds for the different adjacencies are displayed in Figure 10. For the grid adjacency, saddlepoints are more costly to find; but for the row-column adjacency, local optima are more costly. It would be interesting to find a general reason for the opposing behavior of the two neighborhoods.

5 Acknowledgments

The authors gratefully acknowledge support from ONR grant N00014-88-K-0349 (Llewellyn) and NSF grant 84-ECS-8451032 (Tovey). The authors also thank Adam Rosenberg and Michael Trick for helpful discussions.

References

[1] I. Althöfer and K. Koschnick, On the Deterministic Complexity of Searching Local Maxima, manuscript, Universität Bielefeld West- Germany (1989), submitted to Disc. Appl. Math.

[2] D. Hausmann and B. Korte, Lower bounds on the worst-case complexity of some oracle algorithms, Discrete Math. **24** (1978) 261–276.

[3] R. Lipton, D. Rose, and R. Tarjan, Generalized nested Disssection, SIAM J. Numer. Anal. **16** (1979) 346–358.

[4] D. Llewellyn, C. Tovey, and M. Trick, Local optimization on graphs, Disc. Appl. Math. **23** (1989) 157–178.

[5] G. Nemhauser and L. Wolsey, Best algorithms for approximating the maximum of a submodular set function, Math. Oper. Res. **3** (1978) 177–188.

2-Lattice Polyhedra: Duality

Submitted to JCT 7/92

Shiow-yun Chang
Donna C. Llewellyn
John H. Vande Vate
ISyE
Georgia Institute of Technology
Atlanta, Georgia 30332-0205

September 4, 1992

Abstract

In this paper we introduce a class of lattice polyhedra, called 2-Lattice polyhedra. Examples of 2-Lattice polyhedra include bipartite matching polyhedra, the intersection of two integral polymatroids, the connected polyhedron of an undirected graph, and the perfectly matchable subgraph polytope of a bipartite graph. We show that the maximum cardinality of a vector in a 2-Lattice polyhderon is equal to the minimum capacity of a cover. Special cases of this result include König's Theorem, Menger's Theorem, Dilworth's Theorem, and Edmonds' Theorem for cardinality matroid intersection and polymatroid intersection. We show that the collection of minimum covers contains an upper semi-lattice. For special classes of 2-Lattice polyhedra, called Matching 2-Lattice polyhedra, we provide a characterization of the largest member in the family of nested covers in terms of maximum cardinality vectors in the polyhedron.

1 Introduction

Let L be a finite set of elements (called lines) and let Γ be a finite lattice with partial order $(\Gamma, \preceq)$ which induces meet operation $\wedge$ and join operation $\vee$. Let $\beta : \Gamma \mapsto Z$ be submodular and, for each element $\ell \in L$, let $\alpha_\ell : \Gamma \mapsto Z$ be supermodular. Given $S \in \Gamma$ and $x \in R_+^{|L|}$, let $\alpha(S)x = \sum(\alpha_\ell(S)x(\ell) : \ell \in L)$. Then

$$\{x \in R^{|L|} : \alpha(S)x \leq \beta(S) \text{ for each } S \in \Gamma\} \tag{1.1}$$

is a *lattice polyhedron*. Lattice polyhedra were introduced by Hoffman and Schwartz [15] and further studied by Johnson [16], Hoffman [13], Gröflin and Hoffman [11], and Grishuhin [10]. We investigate a special class of lattice polyhedra we call 2-Lattice polyhedra.

Here we consider those lattice polyhedra in which we allow Γ to be infinite, but require a finite bound on the length of chains in Γ. This ensures that Γ is a complete lattice and includes, for example, the lattice of linear subspaces of a finite dimensional vector space. We further require that for each $\ell \in L$, α_ℓ is not only supermodular, but also non-decreasing and maps Γ into $\{0, 1, 2\}$. The set

$$P(\alpha, \beta) = \{x \in R_+^{|L|} : \alpha(S)x \leq \beta(S) \text{ for each } S \in \Gamma\},$$

is called a 2-Lattice polyhedron and each vector $x \in P(\alpha, \beta)$ is called a *2-Lattice vector*. Examples of 2-Lattice polyhedra include bipartite matching polyhedra [14, 18], the intersection of two integral polymatroids [8], the connected polyhedron of an undirected graph [12], and the perfectly matchable subgraph polytope of a bipartite graph [1].

A *cover* is a pair (S, T) of (possibly identical) members of Γ such that

$$\alpha_\ell(S) + \alpha_\ell(T) \geq 2 \text{ for each } \ell \in L.$$

A cover may also be a single element T of Γ (we denote this kind of cover by $(*, T)$) such that

$$\alpha_\ell(T) \geq 2 \text{ for each } \ell \in L.$$

The capacity of a cover (S, T), denoted $\beta(S, T)$, is

$$1/2[\beta(S) + \beta(T)]$$

while the capacity of a cover $(*, T)$, denoted $\beta(*, T)$ is

$$1/2\beta(T).$$

In this paper we consider the relationship between the problem of finding a maximum cardinality 2-Lattice vector:

$$\begin{aligned} \max & \sum_{\ell \in L} x(\ell) \\ \text{s.t } \alpha(S)x & \leq \beta(S) \text{ for each } S \in \Gamma \\ x & \geq 0 \end{aligned} \tag{1.2}$$

and the dual problem:

$$\begin{aligned} \min & \sum_{S \in \Gamma} y(S)\beta(S) \\ \text{s.t. } & \sum_{S \in \Gamma} y(S)\alpha_\ell(S) \geq 1 \quad \text{for each } \ell \in L \\ & y \geq 0 \end{aligned} \tag{1.3}$$

We show that the maximum cardinality of a 2-Lattice vector is the minimum capacity of a cover. Special cases of this result include König's Theorem [17], Menger's Theorem [20], Dilworth's Theorem [6], and Edmonds' Theorem for cardinality matroid intersection and polymatroid intersection [8]. We also show that the set of minimum covers contains an upper semi-lattice.

This paper focuses on the relationships between the the linear programs (1.2) and (1.3), not on the integrality of extreme solutions to (1.2). We refer to $\sum_{\ell \in L} x(\ell)$ as the "cardinality" of a vector x even though x may not be integral. In fact, we only establish the half-integrality of extreme points of (1.2). In many cases, such as bipartite matching and matroid intersection, the polyhedron is known to have integral extreme points, whereas in others, most notably non-bipartite matching, the extreme points are not integral, but the polyhedron does have Chvátal rank 1 [5]. Our ultimate purpose is to characterize those 2-Lattice polyhedra that share this rank 1 property.

Vande Vate [25] has already shown that 2-Lattice polyhedra have half-integral extreme points and that their extreme points correspond to extreme points of related non-bipartite matching problems. Unfortunately, this correspondence by itself is not enough to ensure that 2-Lattice polyhedra have Chvátal rank 1. Thus, we turn to the relationship between α, β and the convex hull of integral 2-Lattice vectors.

All of the examples of 2-Lattice polyhedra relate α and β in some way. We capture these relationships with the following general conditions. First, let $\mathcal{E}$ be a (possibly infinite) set, and let L be a finite subset of $2^{\mathcal{E}}$ (generally chosen to be a collection of pairs from $\mathcal{E}$). We also require that Γ include the empty set and be partially ordered by set containment. In this way, we may associate with each set $S \subseteq \mathcal{E}$ the smallest member, $\sigma(S)$, of Γ containing S. We further require β to be normalized, non-decreasing and satisfy $\beta(\sigma\{e\}) = 1$ for each $e \in \mathcal{E}$, and $\beta(\sigma\{\ell\}) = 2$ for each $\ell \in L$. Finally, we model the relationship between α and β via the condition $\alpha_\ell(S) = \beta(\sigma(\ell) \wedge S)$ for each $\ell \in L$ and $S \in \Gamma$. It is easy to

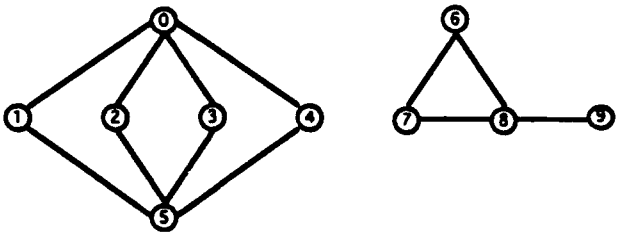

Figure 1.1: Example

see that α_ℓ is normalized and non-decreasing. It is also straightforward to prove (see [25]) that α_ℓ is supermodular. We call the resulting 2-Lattice polyhedra Matching 2-Lattice polyhdera.

When Γ is the family of all subsets of a finite set $\mathcal{E}$ and L is a partition of $\mathcal{E}$ into pairs we refer to the Matching 2-Lattice polyhedra $P(\alpha, \beta)$ as an *incidence 2-Lattice* polyhedron (note that in this setting , $\alpha_\ell : \Gamma \rightarrow \{0, 1, 2\}$ is defined by $\alpha_\ell(S) = |S \cap \ell|$). Integral incidence 2-Lattice polyhedra include bipartite matching polytopes [14, 18], network flow polyhedra [9], and the intersection of two matroids [8]. Incidence 2-Lattice polyhedra have also been studied in the context of non-bipartite matching [7].

Example 1 Consider the cycle matroid of the graph shown in Figure 1. The partition L is given by $L = \{\ell_1, \ell_2, \ell_3, \ell_4, \ell_5, \ell_6\}$, where $\ell_1 = \{(0,1),(1,5)\}$, $\ell_2 = \{(0,2),(2,5)\}$, $\ell_3 = \{(0,3),(3,5)\}$, $\ell_4 = \{(0,4),(4,5)\}$, $\ell_5 = \{(6,7),(6,8)\}$, $\ell_6 = \{(7,8),(8,9)\}$.

Then, $P(\alpha, \beta)$ is the set of $x \in R^6_+$ satisfying

$$\begin{array}{rl} x_i \leq 1 & \text{for } i = 1, 2, \ldots, 6 \\ 2x_i + 2x_j \leq 3 & \text{for } i, j \in \{1, \ldots, 4\}, i \neq j \\ 2x_i + 2x_j + 2x_k \leq 4 & \text{for } i, j, k \in \{1, \ldots, 4\}, i \neq j \neq k \\ 2x_1 + 2x_2 + 2x_3 + 2x_4 \leq 5 & \\ 2x_5 + x_6 \leq 2 & \end{array}$$

Despite the significant successes to date, the formulation of a combinatorial problem via an incidence 2-Lattice polyhedron is not always the best available to us. We can, for instance, improve the incidence formulation in Example 1 via the following matroid formulation. When β is the rank function of a matroid M defined on $\mathcal{E}$, L is a partition of $\mathcal{E}$ into pairs, and Γ is the lattice of flats or closed subsets in M, we refer to $P(\alpha, \beta)$ as a *matroid 2-Lattice* polyhedron.

Example 2 Consider once again the graph of Figure 1. The flats of the cycle matroid of the first connected component consisting of lines ℓ_1, ℓ_2, ℓ_3 and ℓ_4 are: the empty flat; single elements; pairs of elements; pairs of lines; sets of three elements, one from each of three lines; sets of one element from each line; sets

of three lines; and $\{\ell_1, \ell_2, \ell_3, \ell_4\}$. The flats of the cycle matroid of the second connected component consisting of lines ℓ_5 and ℓ_6 are: the empty flat; single elements; sets consisting of $(8,9)$ with another element; $\{(6,7),(6,8),(7,8)\}$; and $\{\ell_5, \ell_6\}$. The flats of this matroid are the combinations of two sets, one from each connected component. Under the matroid formulation $P(\alpha, \beta)$ is the set of $x \in R^6_+$ satisfying

$$\begin{array}{rl} x_i \leq 1 & \text{for } i = 1, 2, \ldots, 6 \\ 2x_i + 2x_j \leq 3 & \text{for } i, j \in \{1, \ldots, 4\}, i \neq j \\ 2x_i + 2x_j + 2x_k \leq 4 & \text{for } i, j, k \in \{1, \ldots, 4\}, i \neq j \neq k \\ 2x_1 + 2x_2 + 2x_3 + 2x_4 \leq 5 & \\ x_5 + x_6 \leq 1 & \end{array}$$

Note that this formulation has the same integral solutions as that of Example 1, but has cut off all extreme points with $x_5 = \frac{1}{2}$ and $x_6 = 1$. For example, it has cut off the extreme points $(0, 0, 0, 0, \frac{1}{2}, 1)$ and $(\frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, \frac{1}{2}, 1)$.

When the matroid is linear and a representation is available, we can do still better than the matroid formulation via the following linear formulation. Let A be a rational matrix and let V denote the linear subspace spanned by the columns of A. If L is a collection of pairs of columns of A, Γ is the lattice of linear subspaces of V and, for each $S \in \Gamma$, $\beta(S)$ denotes the linear rank of S then we refer to $P(\alpha, \beta)$ as a *linear 2-Lattice* polyhedron.

Example 3 We once again consider the graph of Figure 1. Under the linear Matching 2-Lattice formulation, $P(\alpha, \beta)$ is given by the set of $x \in R^6_+$

$$\begin{array}{c} x_1 + x_2 + x_3 + x_4 \leq 1 \\ x_5 + x_6 \leq 1 \end{array}$$

Notice that this is in fact the convex hull of integral solutions to the polyhedron defined in Example 1.

Section 2 gives notation and preliminaries. The first main result of this paper, the min-max theorem for 2-Lattice polyhedra, is proved in Section 3. This section also shows that the family of minimum covers of a 2-Lattice polyhedra contains an upper semi-lattice. In Section 4, we show the second main result of this paper: a characterization of the largest member in the family of nested minimum covers for Matching 2-Lattice polyhedra in terms of maximum cardinality Matching 2-Lattice vectors.

2 Preliminaries

In this section we define notation and present some background results.

Definition 1 Given $x \in R_+^{|L|}$ and a subset S of L, define

$$x_S(\ell) = \begin{cases} x(\ell) & \text{if } \ell \in S \\ 0 & \text{otherwise} \end{cases}$$

Definition 2 For $S, T \in \Gamma$, the rank of T contract S, denoted $\beta(T/S)$, is defined to be $\beta(T \vee S) - \beta(S)$.

In a linear matroid, contraction corresponds to orthogonal projection.

Definition 3 A collection of sets is called an *upper semi-lattice* if it is closed under a join operation.

Definition 4 The *support* of vector $w \in R^n$, denoted by $supp(w)$, is the set $\{i \in [1, 2, \ldots, n] : w_i > 0\}$.

Given a 2-Lattice vector x, the collection of members in Γ tight with respect to x is denoted by $\Gamma(x) = \{S \in \Gamma : \alpha(S)x = \beta(S)\}$. The following lemma shows that $\Gamma(x)$ is a sublattice of Γ.

Lemma 2.1 *Let x be a 2-Lattice vector and suppose S and S' are in $\Gamma(x)$, then $S \vee S'$ and $S \wedge S'$ are in $\Gamma(x)$.*

Proof.

$$\begin{aligned} \beta(S \vee S') + \beta(S \wedge S') &\geq \alpha(S \vee S')x + \alpha(S \wedge S')x \\ &\geq \alpha(S)x + \alpha(S')x \\ &= \beta(S) + \beta(S') \\ &\geq \beta(S \vee S') + \beta(S \wedge S') \end{aligned}$$

□

Since $\Gamma(x)$ is a sublattice of complete lattice, Γ, it has a largest member.

Definition 5 For each 2-Lattice vector x, the largest member of $\Gamma(x)$ is called the *closure* of x and is denoted by $cl(x)$.

The following corollary is an immediate consequence of Lemma 2.1 and will prove useful in arguing that certain vectors $x \in R_+^{|L|}$ are 2-Lattice vectors.

Corollary 2.2 *Let $x \in R_+^{|L|}$ and suppose Z and Z' are members of Γ such that*

$$\begin{aligned} \alpha(Z)x &> \beta(Z), \\ \alpha(Z')x &= \beta(Z') \text{ and} \\ \alpha(Z \wedge Z')x &\leq \beta(Z \wedge Z'), \end{aligned}$$

then $\alpha(Z \vee Z')x > \beta(Z \vee Z')$.

Gröflin and Hoffman [11] demonstrated the following property of lattice polyhedra. (Actually, Gröflin and Hoffman restricted the range of α_ℓ to $\{-1, 0, 1\}$. Nonetheless, their proof applies here as well.)

Theorem 2.3 (Gröflin and Hoffman) *Each extreme point x^* of a lattice polyhedron*

$$\{x \in R_+^{|L|} : \alpha(S)x \leq \beta(S) \text{ for each } S \in \Gamma\}$$

is the unique solution to a system of linear equations:

$$\begin{aligned} \alpha(S_i)x &= \beta(S_i) \quad \text{for } i = 1, \ldots, t \text{ and} \\ x(\ell) &= 0 \quad \text{for } \ell \in N \subseteq L, \end{aligned}$$

where $S = \{S_i : i \in [1, \ldots, t]\}$ is a chain in $(\Gamma, \preceq)$, i.e., $S_1 \prec S_2 \prec \ldots \prec S_t$. □

Using the structure of the bases of the fractional matching polytope of a graph, we are able to describe the structure of extreme 2-Lattice vectors. In particular, Vande Vate (Theorem 2.5 proven in [25]) provides a mechanism for describing extreme 2-Lattice vectors in terms of perfect fractional matchings of graphs.

Given a graph $G = (V, E)$ and an integer vector $b \in R^{|V|}$, the *perfect fractional b-matching polytope* of G, denoted $FP(G, b)$, is:

$$\{x \in R_+^{|E|} : \sum(d_e(v)x(e) : e \in E) = b(v) \text{ for each } v \in V\}.$$

Here, $d_e(v)$ is the degree of edge e at vertex v. As the graph G may have loops, $d_e(v) \in \{0, 1, 2\}$ and as the graph G may have spurs, $\sum(d_e(v) : v \in V) \in \{1, 2\}$. Letting D be the $|V| \times |E|$ matrix with elements $d_e(v)$, $FP(G, b)$ may be written as:

$$FP(G, b) = \{x \in R_+^{|E|} : Dx = b\}$$

Each vector $x \in FP(G, b)$ is a *perfect fractional b-matching* (or, more briefly, a fractional matching) of G.

Chen [4] (also Balinski and Spielberg [2], Trotter [24], Nemhauser and Trotter [21], Pulleyblank [22] and Bartholdi and Ratliff [3]) described the bases of D in terms of the subgraphs induced by the corresponding edges of G.

A subset T of edges is a *bloom* if the subgraph induced by the edges in T is connected, contains exactly one cycle and that cycle has an odd number of edges. In the following theorem, each edge in the graph G is identified with the corresponding column in the matrix D. If G has spurs, we add a distinguished vertex called the *root* incident to each spur edge.

Theorem 2.4 (Chen) *Suppose D is the incidence matrix of a connected graph G. A subset T of columns is a base of D if and only if T is a maximal set of edges such that each component of the subgraph (V, T) is either a tree or a bloom. The component containing the root must be a tree.* □

In light of Theorem 2.4, a set T of edges in a graph G is called a *base* of G if the corresponding columns form a base of the incidence matrix D. Theorem 2.5 extends Theorem 2.4 to the 2-Lattice polyhedron via the following association between extreme 2-Lattice vectors and perfect fractional b-matchings.

By Theorem 2.3, each extreme 2-Lattice vector x^* is defined by a subset N of L and a family $S = \{S_i : i \in [1, \ldots, t]\}$ of members of Γ with $S_1 \prec S_2 \prec \ldots \prec S_t$. For ease of argument and presentation, we form a new complete lattice, Γ^* by appending a new smallest element, $*$, to Γ and defining $\beta(*) = 0$ and $\alpha_\ell(*) = 0$ for each $\ell \in L$. (Note that as a smallest element in Γ^*, $S \wedge * = *$ and $S \vee * = S$ for each $S \in \Gamma$. Further, since $\alpha_\ell(*) = \beta(*) = 0$ it is clear that α_ℓ is supermodular and β is submodular on Γ^*).

The pair (S, N) induces a graph, denoted $G(S, L \setminus N)$, defined as follows. For each $S_i \in S$, there is a vertex S_i in $G(S, L \setminus N)$ and for each line $\ell \in L \setminus N$ there is an edge ℓ in $G(S, L \setminus N)$. Let $S_0 = *$. The edge ℓ is incident to vertex S_i if $\alpha_\ell(S_i) - \alpha_\ell(S_{i-1}) = 1$ and is loop at vertex S_i if $\alpha_\ell(S_i) - \alpha_\ell(S_{i-1}) = 2$.

Theorem 2.5 (Vande Vate) *A 2-Lattice vector x^* is extreme if and only if there is a subset N of L and a family $S = \{S_i : i \in [1, \ldots, t]\}$ of members of Γ with $S_1 \prec S_2 \prec \ldots \prec S_t$ such that:*

1. *$x^*(\ell) = 0$ for each $\ell \in N$,*
2. *$L \setminus N$ is a base of $G(S, L \setminus N)$, and*
3. *The projection of x^* onto the components indexed by lines in $L \setminus N$ is the unique, perfect fractional b-matching in $G(S, L \setminus N)$, where $b(S_i) = \beta(S_i) - \beta(S_{i-1})$ for each $i \in [1, \ldots, t]$.*

Corollary 2.6 *Each extreme 2-Lattice vector is half-integral.*

3 A Min-Max Formula

Theorem 3.1 develops a min-max formula for the maximum cardinality of a 2-Lattice vector. This min-max formula generalizes König's Theorem [17], Menger's Theorem [20], Dilworth's Theorem [6], and Edmonds' Theorem for cardinality matroid intersection and polymatroid intersection [8].

Theorem 3.1 *The maximum cardinality of a 2-Lattice vector is the minimum capacity of a cover.*

Proof. To see that the maximum cardinality of a 2-Lattice vector is at most the minimum capacity of a cover, observe that for any cover (S, T), the solution $y(S) = y(T) = 1/2$ is dual feasible and has objective value $\beta(S, T)$.

To prove that the maximum cardinality of a 2-Lattice vector equals the minimum capacity of a cover, we show that there is an optimum solution y^* to the dual problem such that:

1. $supp(y^*)$ forms a chain in $(\Gamma, \preceq)$.

2. y^* is half-integral,

3. $y^*(S) > 0$ for at most two members $S \in \Gamma$.

First, to see that there is an optimum solution y^* to the dual problem satisfying (1) we employ an argument similar to that of Hoffman and Schwatrz [15], but modified to accommodate an infinite lattice Γ. Consider an optimal dual solution $\bar{y}$ with finite support (e.g. each extreme point optimal solution has finite support). If $supp(\bar{y})$ forms a chain in $(\Gamma, \preceq)$, we are done. Otherwise, define a complete order $\prec'$ on $supp(\bar{y})$ that is consistent with the partial order $\preceq$. We argue that $\bar{y}$ can be converted into a dual solution y^* such that $supp(y^*)$ forms a chain in $(\Gamma, \preceq)$ as follows.

Let $S_0 = *$ and index the elements of $supp(\bar{y})$ so that

$$S_0 \prec' S_1 \prec' S_2 \prec' \cdots S_t.$$

Define $i = i_{\bar{y}}$ to be the smallest index such that $S_{i-1} \not\preceq S_i$ and $j = j_{\bar{y}}$ to be the smallest index such that $S_j \not\preceq S_i$. Consider the dual solution $\tilde{y}$ such that

$$\tilde{y}(S) = \begin{cases} \bar{y}(S) - \epsilon & \text{if } S \in \{S_i, S_j\} \\ \bar{y}(S) + \epsilon & \text{if } S \in \{S_i \vee S_j, S_i \wedge S_j\} \\ \bar{y}(S) & \text{otherwise,} \end{cases}$$

where $\epsilon = \min\{\bar{y}(S_i), \bar{y}(S_j)\}$. Since

$$\alpha_\ell(S_i \vee S_j) + \alpha_\ell(S_i \wedge S_j) \geq \alpha_\ell(S_i) + \alpha_\ell(S_j)$$

for each line $\ell \in L$, $\tilde{y}$ is dual feasible. Further, since

$$\beta(S_i \vee S_j) + \beta(S_i \wedge S_j) \leq \beta(S_i) + \beta(S_j),$$

$$\sum_{S \in \Gamma} \tilde{y}(S)\beta(S) \leq \sum_{S \in \Gamma} \bar{y}(S)\beta(S).$$

So, the (dual) objective value of $\tilde{y}$ is no worse than that of $\bar{y}$.

Note that the chain $S_0 \preceq S_1 \preceq \cdots S_i \wedge S_j \preceq S_j \preceq \cdots S_{i-1}$ in $(\Gamma, \preceq)$ grows with each successive revision of this kind. Since there is a finite upper bound on the length of any chain in Γ, this process must ultimately terminate with a dual solution y^* such that $supp(y^*)$ is a chain in $(\Gamma, \preceq)$.

Now, to see that y^* satisfies (2), let $\mathcal{S} = \{S_i : i = 1, \ldots, t\}$ be a nested family of members of Γ and N a subset of L such that y^* is the unique solution to the system:

$$\sum_{S_i \in \mathcal{S}} y(S_i)\alpha_\ell(S_i) = 1 \text{ for each } \ell \in L \setminus N \tag{3.4}$$

Let y' be the unique solution to the system:

$$\sum_{S_i \in S} y(S_i)(\alpha_\ell(S_i) - \alpha_\ell(S_{i-1})) = 1 \text{ for each } \ell \in L \setminus N \tag{3.5}$$

Then y' is the unique solution to the system $yA = 1$, where A is the node-edge incidence matrix of the basis graph $G(S, L \setminus N)$, i.e.,

$$y'(S_i) = \begin{cases} 1/2 & \text{if there is no path in } G(S, L \setminus N) \text{ from the root to } S_i, \\ 1 & \text{if there are an odd number of edges on the path in} \\ & G(S, L \setminus N) \text{ from the root to } S_i, \text{ and} \\ 0 & \text{if there are an even number of edges on the path in} \\ & G(S, L \setminus N) \text{ from the root to } S_i. \end{cases}$$

And, we may compute y^* as follows:

$$\begin{aligned} y^*(S_i) &= y'(S_i) - y'(S_{i+1}) \text{ for } i = 1, \ldots, t-1, \text{ and} \\ y^*(S_t) &= y'(S_t). \end{aligned}$$

It follows immediately that y^* is half-integral.

Finally, to see that y^* has at most two non-zero components observe that since y^* is dual feasible, it is non-negative. Thus, the corresponding vector y' must be of the form

$$y'(S_i) = \begin{cases} 1 & \text{for } i = 1, \ldots i_1 \\ 1/2 & \text{for } i = i_1 + 1, \ldots, i_2 \\ 0 & \text{for } i = i_2 + 1, \ldots, t. \end{cases}$$

It follows that y^* has at most two non-zero components and $\sum_{S_i \in S} y^*(S_i) \in \{1/2, 1\}$. If y^* has exactly two non-zero components S and T, then $y^*(S) = y^*(T) = 1/2$ and (S, T) is a minimum cover. If y^* has only one non-zero component S, then either $y^*(S) = 1$, in which case (S, S) is a minimum cover, or $y^*(S) = 1/2$ in which case $(*, S)$ is a minimum cover. □

Definition 6 A cover (S, T) with $S \preceq T$ is called a *nested cover.*

The following lemma shows that we may associate a nested cover with each minimum cover and hence that there is always a nested minimum cover.

Lemma 3.2 *If (S, T) is a minimum cover then $(S \wedge T, S \vee T)$ is a nested minimum cover.*

Proof. For each $\ell \in L$, $\alpha_\ell(S \wedge T) + \alpha_\ell(S \vee T) \geq \alpha_\ell(S) + \alpha_\ell(T) \geq 2$. Therefore, $(S \wedge T, S \vee T)$ is a cover. Since $\beta(S \vee T) + \beta(S \wedge T) \leq \beta(S) + \beta(T)$, it follows that $(S \wedge T, S \vee T)$ is a minimum cover. □

Corollary 3.3 *The maximum cardinality of a 2-Lattice vector is the minimum capacity of a nested cover.*

We present Edmond's duality theorem for cardinality matroid intersection as a special case of Theorem 3.1.

Corollary 3.4 *Let $\mathbf{M}_1$ be a matroid with rank function r_1 and let $\mathbf{M}_2$ be a matroid with rank function r_2 both defined on the same ground set E. Then the maximum cardinality of an intersection in $\mathbf{M}_1$ and $\mathbf{M}_2$ is*

$$\min_{S \subseteq E} r_1(S) + r_2(E \setminus S).$$

Proof. The matroid intersection polyhedron

$$P = \{x \in R_+^E : x(S) \leq r_1(S) \text{ and } x(S) \leq r_2(S) \text{ for each } S \subseteq E\}$$

is equivalent to the 2-Lattice polyhedron

$$\{x \in R_+^L : \alpha(S)x \leq \beta(S) \text{ for each } S \subseteq \mathcal{E}\}$$

where

- $\mathcal{E}$ consists of two copies E and E' of E,
- L consists of the lines $\{e, e'\}$ with an element from E and its copy in E',
- For each $\ell \in L$ and $S \subseteq \mathcal{E}$, $\alpha_\ell(S) = |\ell \cap S|$, and
- For each $S \in \mathcal{E}$, $\beta(S) = r_1(S \cap E) + r_2(S \cap E')$.

Thus, Corollary 3.3 implies that the maximum cardinality of an intersection in $\mathbf{M}_1$ and $\mathbf{M}_2$ is the minimum capacity of a nested cover. Let (S, T) be a minimum capacity nested cover. Define $S_1 = S \cap E$ and $S_2 = S \cap E'$. Similarly, let $T_1 = T \cap E$ and $T_2 = T \cap E'$. Since (S, T) is a nested cover, if $e \notin S_1$, then $e' \in T_2$ and, if $e' \notin S_2$ then $e \in T_1$. Thus,

$$r_1(S_1) + r_2(E \setminus S_1) \leq r_1(S_1) + r_2(T_2)$$

and

$$r_1(E \setminus S_2) + r_2(S_2) \leq r_1(T_1) + r_2(S_2).$$

It is easy to establish that for each $x \in P$ and $S \subseteq E$

$$\sum_{e \in E} x(e) \leq r_1(S) + r_2(E \setminus S).$$

Since each maximum cardinality $x \in P$ satisfies

$$\sum_{e \in E} x(e) = \beta(S, T) = 1/2[r_1(S_1) + r_2(T_2) + r_1(T_1) + r_2(S_2)],$$

It follows that $\sum_{e \in E} x(e) = r_1(S_1) + r_2(E \setminus S_1) = r_1(E \setminus S_2) + r_2(S_2)$ and hence that the maximum cardinality of an intersection in $\mathbf{M}_1$ and $\mathbf{M}_2$ is equal to $\min_{S \subseteq E} r_1(S) + r_2(E \setminus S)$. □

When the 2-Lattice polyhedron is known to have integral extreme points, we may restrict attention to integer 2-Lattice vectors in Theorem 3.1. In the case of matroid intersection, it is easy to verify that for each family $\mathcal{S} = \{S_i : i \in [1, \ldots, t]\}$ of members of Γ with $S_1 \prec S_2 \prec \ldots \prec S_t$, $G(\mathcal{S}, L)$ is bipartite and hence, as is well known, the extreme points of the matroid intersection polyhedron are integral.

Note that the notation used in the proof of Theorem 3.1 is consistent with our construction of the lattice Γ^*. Therefore we henceforth refer to all covers as (S, T) with the understanding that S may be $*$.

We can use our linear programming formulation to further characterize minimum capacity covers.

Corollary 3.5 *For each minimum cover (S, T) and maximum 2-Lattice vector x,*

- $\alpha(S)x = \beta(S)$
- $\alpha(T)x = \beta(T)$ *and,*
- *if* $\alpha_\ell(S) + \alpha_\ell(T) > 2$, *then* $x(\ell) = 0$

Proof. By complementary slackness. □

Given a 2-Lattice polyhedron, let Ω be the collection of all maximum cardinality 2-Lattice vectors and Ω_{ext} be the collection of all extreme maximum cardinality 2-Lattice vectors.

Corollary 3.6 *For each minimum cover (S, T),*

$$S, T \preceq \wedge(cl(x) : x \in \Omega) \preceq \wedge(cl(x) : x \in \Omega_{ext})$$

Shapley and Shubik [23] showed that the collection of optimal dual solutions to a bipartite matching problem form a lattice. The same result holds for cardinality matroid intersection. In particular, if $(S, E \setminus S)$ and $(S', E \setminus S')$ are dual solutions in the sense of Corollary 3.4 to a matroid intersection problem, then so are $(S \cap S', E \setminus (S \cap S'))$ and $(S \cup S', E \setminus (S \cup S'))$. We show that the set of nested minimum covers for a 2-Lattice polytope forms an upper semi-lattice. It remains an open question whether these covers in fact form a lattice.

Lemma 3.7 *If (S_1, T_1) and (S_2, T_2) are nested minimum covers, then $(S_1 \wedge S_2, T_1 \vee T_2)$ and $(S_1 \vee S_2, T_1 \wedge T_2)$ are minimum covers.*

Proof. We first show that $(S_1 \wedge S_2, T_1 \vee T_2)$ and $(S_1 \vee S_2, T_1 \wedge T_2)$ are covers. Since (S_1, T_1) and (S_2, T_2) are covers and α_ℓ is supermodular for each $\ell \in L$,

$$\begin{aligned} \alpha_\ell(S_1 \wedge S_2) + \alpha_\ell(S_1 \vee S_2) + \alpha_\ell(T_1 \wedge T_2) + \alpha_\ell(T_1 \vee T_2) &\geq \\ \alpha_\ell(S_1) + \alpha_\ell(S_2) + \alpha_\ell(T_1) + \alpha_\ell(T_2) &\geq 4. \end{aligned}$$

And so, we need only consider the cases in which $\alpha_\ell(S_1 \vee S_2) + \alpha_\ell(T_1 \wedge T_2)$ or $\alpha_\ell(S_1 \wedge S_2) + \alpha_\ell(T_1 \vee T_2)$ is strictly greater than 2.

Case 1. If $\alpha_\ell(S_1 \vee S_2) + \alpha_\ell(T_1 \wedge T_2) > 2$, either $\alpha_\ell(S_1 \vee S_2)$ or $\alpha_\ell(T_1 \wedge T_2) = 2$. However, since (S_1, T_1) and (S_2, T_2) are nested,

$$(S_1 \wedge S_2) \preceq (S_1 \vee S_2) \preceq (T_1 \vee T_2)$$

and

$$(S_1 \wedge S_2) \preceq (T_1 \wedge T_2) \preceq (T_1 \vee T_2).$$

So, $\alpha_\ell(T_1 \vee T_2) = 2$; proving that $\alpha_\ell(S_1 \wedge S_2) + \alpha_\ell(T_1 \vee T_2) \geq 2$.

Case 2. If $\alpha_\ell(S_1 \wedge S_2) + \alpha_\ell(T_1 \vee T_2) > 2$, then $\alpha_\ell(S_1 \wedge S_2) \geq 1$ and so,

$$1 \leq \alpha_\ell(S_1 \wedge S_2) \leq \alpha_\ell(S_1 \vee S_2).$$

Similarly,

$$1 \leq \alpha_\ell(S_1 \wedge S_2) \leq \alpha_\ell(T_1 \wedge T_2);$$

proving that $\alpha_\ell(S_1 \vee S_2) + \alpha_\ell(T_1 \wedge T_2) \geq 2$.

Thus, $\alpha_\ell(S_1 \vee S_2) + \alpha_\ell(T_1 \wedge T_2) \geq 2$ and $\alpha_\ell(S_1 \wedge S_2) + \alpha_\ell(T_1 \vee T_2) \geq 2$ for each $\ell \in L$, i.e., $(S_1 \wedge S_2, T_1 \vee T_2)$ and $(S_1 \vee S_2, T_1 \wedge T_2)$ are covers.

Since $(S_1 \wedge S_2, T_1 \vee T_2)$ and $(S_1 \vee S_2, T_1 \wedge T_2)$ are covers and (S_1, T_1) and (S_2, T_2) are minimum covers

$$\beta(S_1 \wedge S_2) + \beta(T_1 \vee T_2) \geq \beta(S_1) + \beta(T_1)$$

and

$$\beta(S_1 \vee S_2) + \beta(T_1 \wedge T_2) \geq \beta(S_2) + \beta(T_2).$$

But, since β is submodular,

$$\beta(S_1 \wedge S_2) + \beta(S_1 \vee S_2) + \beta(T_1 \wedge T_2) + \beta(T_1 \vee T_2) \leq \beta(S_1) + \beta(S_2) + \beta(T_1) + \beta(T_2).$$

Thus, we must have equality throughout. □

Let $\mathcal{C}$ be the collection of all nested minimum covers. We show that $\mathcal{C}$ is a upper semi-lattice with partial order defined by $(S, T) \preceq (S', T')$ if

- $T \preceq T'$ and

- $S' \preceq S$.

In fact, we show that the binary operation $\vee_e$ on $\mathcal{C}$ defined by

$$(S,T) \vee_e (S',T') = (S \wedge S', T \vee T')$$

is the join operation in $\mathcal{C}$.

Lemma 3.8 *$\mathcal{C}$ is an upper semi-lattice.*

Proof. By Lemma 3.2 and Lemma 3.7, $(S \wedge S', T \vee T')$ is a nested minimum cover. It is easy to vertify that this is also the least upper bound of (S,T) and (S',T'). Thus, $\mathcal{C}$ is an upper semi-lattice. □

The following example shows that $\mathcal{C}$ need not be a lattice. Consider the incidence 2-Lattice polyhedra on $\mathcal{E} = \{e,f\}$ with the single line $\ell = \{e,f\}$ and $\beta(S)$ defined by $|S|$. The nested minimum covers are $(\emptyset,\mathcal{E})$, $(\{e\},\{e\})$ and $(\{f\},\{f\})$. Clearly $(\emptyset,\mathcal{E})$ is the least upper bound of $(\{e\},\{e\})$ and $(\{f\},\{f\})$, but these two nested covers do not have a common lower bound in $\mathcal{C}$.

The following corollary shows that there is a largest cover in $\mathcal{C}$ and in some sense this cover dominates all others.

Corollary 3.9 *There is a nested minimum cover (S^*,T^*), such that $T \preceq T^*$ and $S^* \preceq S$ for each minimum cover (S,T).*

Proof. Let (S^*,T^*) be any nested minimum cover with the property that no nested minimum cover (S,T) has $T^* \prec T$ or $S \prec S^*$ (since there is a finite bound on the length of any chain in Γ, such a cover exists). Suppose that (S,T) is a minimum cover with $T \not\preceq T^*$ or $S^* \not\preceq S$. By Lemma 3.2 $(S \wedge T, S \vee T)$ is a nested minimum cover. So, by Lemma 3.7, $(S \wedge T \wedge S^*, S \vee T \vee T^*)$ is a nested minimum cover and $T^* \prec S \vee T \vee T^*$ or $S \wedge T \wedge S^* \prec S^*$ contradicting the choice of (S^*,T^*). □

We refer to the nested minimum cover of Corollary 3.9 as the *dominant cover.*

4 The Dominant Cover

The most common lattice polyhedra with $\alpha \in \{0,1,2\}$ include bipartite matching polyhedra and matroid intersection polyhedra. In each case, $\alpha_\ell(S) = |\ell \cap S|$. Here we generalize this relationship between α and β. Let $\mathcal{E}$ be a (possibly infinite) set, and let L be a finite subset of $2^{\mathcal{E}}$ (generally chosen to be a collection of pairs from $\mathcal{E}$).

We require that Γ contain $\emptyset$ and be partially ordered by set containment. Recall that we associate with each set $S \subseteq \mathcal{E}$ the smallest member, $\sigma(S)$, of Γ

containing S. We extend the meet and join operation of Γ to all subsets of $\mathcal{E}$ so that for S and $T \subseteq \mathcal{E}$, $S \wedge T = \sigma(S) \wedge \sigma(T)$ and $S \vee T = \sigma(S \cup T)$.

We further require that β be normalized, non-decreasing and satisfy $\beta(\sigma\{e\}) = 1$ for each $e \in \mathcal{E}$, and $\beta(\sigma\{\ell\}) = 2$ for each $\ell \in L$. Finally, we model the relationship between α and β via the condition $\alpha_\ell(S) = \beta(\ell \wedge S)$ for each $\ell \in L$ and $S \in \Gamma$. It is easy to see that α_ℓ is normalized and non-decreasing. It is also straightforward to prove (see [25]) that α_ℓ is supermodular. We call the resulting 2-Lattice polyhedra Matching 2-Lattice polyhedra.

The following definitions prove useful:

Definition 7 A *base* of a subset, $S \subseteq \mathcal{E}$, is a minimal subset $T \subseteq S$ with $\sigma(T) = \sigma(S)$.

For example, if Γ is the collection of flats in a matroid, then a maximal independent set in S is a base of S. If Γ is the collection of linear subspaces of a vector space a maximal linearly independent set of vectors in S is a base of S.

Definition 8 For $T \in \Gamma$, let $L_T = \{\ell \in L : \alpha_\ell(T) = 1\}$.

Lemma 4.1 shows that given one element of a nested minimum cover, we can characterize the other.

Lemma 4.1 *If (S,T) is a nested minimum cover then $S = \sigma(\{\ell \wedge T : \ell \in L_T\})$ and $T = S \vee \sigma(\{\ell \in L : \alpha_\ell(S) = 0\})$.*

Proof. Since (S,T) is a nested cover, $S' = \sigma(\{\ell \wedge T : \ell \in L_T\}) \subseteq S$. Further, since (S',T) is a cover,

$$\beta(S) + \beta(T) \leq \beta(S') + \beta(T).$$

It follows that $S' = S$.

Similarly, since (S,T) is a nested cover, $T' = S \vee \sigma(\{\ell \in L : \alpha_\ell(S) = 0\}) \subseteq T$ and since (S,T') is a cover,

$$\beta(S) + \beta(T) \leq \beta(S) + \beta(T').$$

It follows that $T' = T$. □

Now we characterize the dominant cover in terms of maximum Matching 2-Lattice vectors.

Let (S,T) be a nested cover and for each $\ell \in L_T$, let $e(\ell) \in \ell \wedge T$. Define the matroid $\mathcal{M}_1(S,T)$ with rank function r_1 on L_T as follows. A set X of lines in L_T is independent in $\mathcal{M}_1(S,T)$ if $\beta(\{e(\ell) : \ell \in X\}) = |X|$.

Define the matroid $\mathcal{M}_2(S,T;\{e\})$ with rank function r_2 on L_T as follows. A set X of lines in L_T is independent in $\mathcal{M}_2(S,T;\{e\})$ if $\beta(X/T \vee \{e\}) = |X|$.

Lemma 4.2 shows that if the maximum cardinality of an intersection in $\mathcal{M}_1$ and $\mathcal{M}_2$ is $\beta(S) - 1$, there is a cover (S',T') with $T \vee e \subseteq T'$.

Lemma 4.2 *If (S,T) is a nested minimum cover and $e \notin T$, then the maximum cardinality of an intersection in $\mathcal{M}_1(S,T)$ and $\mathcal{M}_2(S,T;\{e\})$ is either $\beta(S)$ or $\beta(S)-1$. Futhermore, if the maximum cardinality of an intersection in $\mathcal{M}_1$ and $\mathcal{M}_2$ is $\beta(S)-1$ then there is a minimum cover (S',T') such that $T \vee e \subseteq T'$.*

Proof. The maximum cardinality of an intersection in $\mathcal{M}_1$ and $\mathcal{M}_2$ is bounded by $\beta(S)$. Suppose the maximum cardinality of an intersection in $\mathcal{M}_1$ and $\mathcal{M}_2$ is less than or equal to $\beta(S)-1$, then there is a minimum rank cover (X_1,X_2) of L_T for the matroid intersection problem such that

$$r_1(X_1) + r_2(X_2) \leq \beta(S) - 1,$$

that is,

$$\beta(\{e(\ell) : \ell \in X_1\}) + \beta(X_2/(T \vee e)) \leq \beta(S) - 1$$

and so

$$\beta(\{e(\ell) : \ell \in X_1\}) + \beta(X_2 \vee T \vee e) \leq \beta(S) + \beta(T \vee e) - 1 = \beta(S) + \beta(T).$$

Let $S' = \sigma(\{e(\ell) : \ell \in X_1\})$ and $T' = X_2 \vee T \vee e$. Then (S',T') is a cover of L with $T \vee e \subseteq T'$ and $\beta(S',T') \leq \beta(S,T)$. Since (S,T) is a minimum cover, it follows that (S',T') is a minimum cover and the size of a maximum intersection must be at least $\beta(S)-1$. □

Corollary 4.3 *If (S^*,T^*) is the dominant cover and $e \notin T^*$, then the maximum cardinality of an intersection in $\mathcal{M}_1(S^*,T^*)$ and $\mathcal{M}_2(S^*,T^*;\{e\})$ is $\beta(S^*)$.*

The following two lemmas identify special properties of maximum Matching 2-Lattice vectors and show conditions under which we may combine portions of two Matching 2-Lattice vectors to form a third.

Lemma 4.4 *Let x be a maximum Matching 2-Lattice vector and let (S,T) be a nested minimum cover. Then $x_{L \setminus L_T}$ satisfies*

1. $\alpha(T)x_{L \setminus L_T} = \beta(T/S)$ *and*
2. *for $T' \subseteq T$, $\alpha(T')x_{L \setminus L_T} \leq \beta(T'/S)$*

and x_{L_T} satisfies

3. $\alpha(T)x_{L_T} = \beta(S)$,
4. *for $T' \subseteq T$, $\alpha(T')x_{L_T} \leq \beta(T' \wedge S)$.*

Proof. First, observe that for each line $\ell \in L \setminus L_T$, $\alpha_\ell(T) = 2$. So, if $\alpha_\ell(S) > 0$, $\alpha_\ell(S) + \alpha_\ell(T) > 2$ and, by Corollary 3.5, $x(\ell) = 0$. Thus, $\alpha(S)x_{L \setminus L_T} = 0$. Since $\alpha(S)x = \beta(S)$, it follows that $\alpha(S)x_{L_T} = \beta(S)$.

To see (3), observe that for each $\ell \in L_T$, $\alpha_\ell(T) = \alpha_\ell(S) = 1$. So,

$$\alpha(T)x_{L_T} = \alpha(S)x_{L_T} = \beta(S).$$

To see (1), observe that since

$$\alpha(T)x = \beta(T) = \beta(T \vee S) \text{ and } \alpha(T)x_{L_T} = \beta(S)$$

it follows that

$$\alpha(T)x_{L\setminus L_T} = \beta(T/S).$$

To see (2), observe that for $T' \subseteq T$,

$$\alpha(T' \vee S)x \leq \beta(T' \vee S) \text{ and } \alpha(T' \vee S)x_{L_T} = \beta(S).$$

Thus,

$$\begin{aligned} \alpha(T')x_{L\setminus L_T} &\leq \alpha(T' \vee S)x_{L\setminus L_T} \\ &= \alpha(T' \vee S)x - \alpha(T' \vee S)x_{L_T} \\ &\leq \beta(T' \vee S) - \beta(S) \\ &= \beta(T'/S). \end{aligned}$$

To see (4), note that for $\ell \in L_T$, $\alpha_\ell(S) = \alpha_\ell(T)$. So,

$$\alpha(T')x_{L_T} = \alpha(T' \wedge S)x_{L_T} \leq \beta(T' \wedge S).$$

□

Lemma 4.5 *Let x and $\tilde{x}$ be Matching 2-Lattice vectors and let (S,T) be a nested minimum cover. If x satisfies (1) and (2) of Lemma 4.4, $\tilde{x}$ satisfies (3) and (4) of Lemma 4.4,*

a. $\beta(T/cl(\tilde{x}_{L_T})) = \beta(T/S)$ *and*

b. $\alpha(cl(\tilde{x}_{L_T}))x_{L\setminus L_T} = 0$,

then $x' = \tilde{x}_{L_T} + x_{L\setminus T}$ is a Matching 2-Lattice vector.

Proof. Suppose x' is not a Matching 2-Lattice vector, then there is a flat $Z \in \Gamma$ such that $\alpha(Z)x' > \beta(Z)$. We first show that we may choose Z to contain $cl(\tilde{x}_{L_T})$.

By condition (b), $\alpha(Z \wedge cl(\tilde{x}_{L_T}))x' = \alpha(Z \wedge cl(\tilde{x}_{L_T}))\tilde{x}_{L_T}$, and since $\tilde{x}_{L_T}$ is feasible $\alpha(Z \wedge cl(\tilde{x}_{L_T}))\tilde{x}_{L_T} \leq \beta(Z \wedge cl(\tilde{x}_{L_T}))$. It follows by Corollary 2.2 that

$$\alpha(Z \vee cl(\tilde{x}_{L_T}))x' > \beta(Z \vee cl(\tilde{x}_{L_T})).$$

Thus, if x' is not a Matching 2-Lattice vector, there is a flat $Z \in \Gamma$ with $cl(\tilde{x}_{L_T}) \subseteq Z$ such that $\alpha(Z)x' > \beta(Z)$.

We next show that we may also assume $T \subseteq Z$.

By conditions (1) and (3)

$$\alpha(T)x' = \alpha(T)\tilde{x}_{L_T} + \alpha(T)x_{L\setminus L_T} = \beta(S) + \beta(T/S) = \beta(T).$$

Further, by conditions (2) and (4)

$$\alpha(Z \wedge T)x' = \alpha(Z \wedge T)\tilde{x}_{L_T} + \alpha(Z \wedge T)x_{L\setminus L_T} \leq \beta(Z \wedge T \wedge S) + \beta((Z \wedge T)/S).$$

Therefore, $\alpha(Z \wedge T)x' \leq \beta(Z \wedge T)$, and so it follows by Corollary 2.2 that $\alpha(Z \vee T)x' > \beta(Z \vee T)$. But,

$$\begin{array}{rcll}
\alpha(Z \vee T)x' & = & \alpha(Z \vee T)\tilde{x}_{L_T} + \alpha(Z \vee T)x_{L\setminus L_T} & \\
& = & \beta(cl(\tilde{x}_{L_T})) + \alpha(Z \vee T)x_{L\setminus L_T} & \text{since } cl(\tilde{x}_{L_T}) \subseteq Z \\
& = & \beta(cl(\tilde{x}_{L_T})) + \alpha(T)x_{L\setminus L_T} & \text{since } supp(x_{L\setminus L_T}) \subseteq T \\
& = & \beta(cl(\tilde{x}_{L_T})) + \beta(T/S) & \text{by (1)} \\
& = & \beta(cl(\tilde{x}_{L_T})) + \beta(T/cl(\tilde{x}_{L_T})) & \text{by (a)} \\
& = & \beta(cl(\tilde{x}_{L_T}) \vee T) & \\
& \leq & \beta(Z \vee T) & \text{since } cl(\tilde{x}_{L_T}) \subseteq Z.
\end{array}$$

This contradicts the existence of Z and proves that x' is a Matching 2-Lattice vector. □

Lemma 4.6 shows that if (S,T) is a nested minimum cover and $e \notin T$, then each $\beta(S)$-intersection in $\mathcal{M}_1(S,T)$ and $\mathcal{M}_2(S,T;\{e\})$ gives rise to a maximum Matching 2-Lattice vector x with $e \notin cl(x)$.

Lemma 4.6 *Let x be a maximum Matching 2-Lattice vector, (S,T) be a nested minimum cover and $e \notin T$. If X is a $\beta(S)$ intersection in $\mathcal{M}_1(S,T)$ and $\mathcal{M}_2(S,T;\{e\})$, then x' defined by*

$$x'(\ell) = \begin{cases} 1 & \text{if } \ell \in X \\ 0 & \text{if } \ell \in L_T \setminus X \\ x(\ell) & \text{otherwise} \end{cases}$$

is a maximum cardinality Matching 2-Lattice vector with $e \notin cl(x')$.

Proof. First, since X is independent in $\mathcal{M}_2(S,T;\{e\})$ and $|X| = \beta(S)$,

$$\beta(X/T \vee \{e\}) = |X| = \beta(S)$$

and so

$$\beta(X \vee T \vee \{e\}) = \beta(S) + \beta(T \vee \{e\}).$$

Second, since X is independent in $\mathcal{M}_1(S,T)$ and $|X| = \beta(S)$, $\{e(\ell) : \ell \in X\}$ is a base of S. Let $\{e(\ell) : \ell \in X\} \cup B$ be a base of T. Then $e \notin \sigma(X \cup B)$ and

$$\begin{aligned}\beta(X/(B \cup \{e\})) &= \beta(X \cup B \cup \{e\}) - \beta(B \cup \{e\}) \\ &= \beta(S) + \beta(T \vee \{e\}) - \beta(B \cup \{e\}) \\ &= 2\beta(S) \\ &= 2|X|.\end{aligned}$$

It follows that x'_{L_T} is a Matching 2-Lattice vector.

We see that x'_{L_T} satisfies (3) of Lemma 4.4 as follows. Since $\alpha_\ell(T) = 1$ for each $\ell \in X$,

$$\alpha(T)x'_{L_T} = |X| = \beta(S).$$

We see that x'_{L_T} satisfies (4) of Lemma as follows. Since $\alpha_\ell(S) = \alpha_\ell(T) = 1$ for each $\ell \in X$, if $T' \subseteq T$,

$$\alpha(T')x'_{L_T} = \alpha(T' \wedge S)x'_{L_T} \leq \beta(T' \wedge S).$$

Since x is a maximum Matching 2-Lattice vector, $x_{L \setminus L_T}$ satisfies conditions (1) and (2) of Lemma 4.4. Thus, to show that x' is a Matching 2-Lattice vector, we need only show that x'_{L_T} and $x_{L \setminus L_T}$ satisfy conditions (a) and (b) of Lemma 4.5.

We see that x'_{L_T} satisfies (a) of Lemma 4.5 as follows. Since

$$cl(x'_{L_T}) \subseteq \sigma(supp(x'_{L_T})) = \sigma(X)$$

and

$$\alpha(\sigma(X))x'_{L_T} = 2|X| = \beta(\sigma(X)),$$

it follows that $cl(x'_{L_T}) = \sigma(X)$. Therefore,

$$\beta(T/cl(x'_{L_T})) = \beta(T/X) = |B| = \beta(T/\sigma(\{e(\ell) : \ell \in X\})) = \beta(T/S).$$

We see that x'_{L_T} and $x_{L \setminus L_T}$ satisfy condition (b) of Lemma 4.5 as follows. Since $supp(x'_{L \setminus L_T}) \subseteq T$ and $cl(x'_{L_T}) = \sigma(X)$,

$$\alpha(cl(x'_{L_T}))x_{L \setminus L_T} = \alpha(X \wedge T)x_{L \setminus L_T}.$$

But $X \wedge T = S$ so

$$\alpha(cl(x'_{L_T}))x_{L \setminus L_T} = \alpha(S)x_{L \setminus L_T} = 0.$$

Thus, by Lemma 4.5, x' is a Matching 2-Lattice vector.

Since $e \notin \sigma(X \cup T)$ and $cl(x') \subseteq \sigma(supp(x')) \subseteq \sigma(X \cup T)$, it follows that $e \notin cl(x')$.

To see that x' is a maximum cardinality Matching 2-Lattice vector, observe that

$$\begin{aligned}
\sum_{\ell \in L} x'(\ell) &= \sum_{\ell \in L_T} x'(\ell) + \sum_{\ell \in L \setminus L_T} x(\ell) \\
&= \beta(S) + \sum_{\ell \in L} x(\ell) - \sum_{\ell \in L_T} x(\ell) \\
&= \beta(S) + \beta(S,T) - \sum_{\ell \in L_T} x(\ell) \\
&\geq \beta(S) + \beta(S,T) - \beta(S) \\
&= \beta(S,T)
\end{aligned}$$

□

Corollary 4.7 *If (S^*, T^*) is the dominant cover, then $T^* \supseteq \cap(cl(x) : x \in \Omega)$*

Proof. By Corollary 4.3, if $e \notin T^*$, then the maximum cardinality of an intersection in $\mathcal{M}_1(S^*, T^*)$ and $\mathcal{M}_2(S^*, T^*; \{e\})$ is $\beta(S^*)$. By Lemma 4.6, there is $x \in \Omega$ such that $e \notin cl(x)$, hence, $e \notin \cap(cl(x) : x \in \Omega)$. Therefore, $T^* \supseteq \cap(cl(x) : x \in \Omega)$ □

Combining Corollary 3.6, Corollary 4.7 and Lemma 4.1, we have the following characterization of the dominant cover in terms of maximum Matching 2-Lattice vectors.

Theorem 4.8 *Let $T^* = \cap(cl(x) : x \in \Omega)$ and $S^* = \sigma(\{\ell \wedge T^* : \ell \in L_{T^*}\})$. Then (S^*, T^*) is the dominant cover.*

The following results refine Lemma 4.6 to extreme maximum Matching 2-Lattice vectors.

Lemma 4.9 *Let (S^*, T^*) be the dominant cover. Then, for each $x \in \Omega_{ext}$*

1. *For each $\ell \in L_{T^*}$, $x(\ell) \in \{0, 1\}$,*
2. *$\beta(T^*/cl(x_{L_{T^*}})) = \beta(T^*/S^*)$, and*
3. *$T^* \wedge cl(x_{L_{T^*}}) = S^*$.*

Proof. For each $x^* \in \Omega_{ext}$, there is a complementary dual solution y^*. Let $S = \{S_i : i = 1, \cdots, t\}$ be a nested family of flats in Γ and N a subset of L such that x^* is the unique solution to the system:

$$\begin{aligned}
\alpha(S_i)x &= \beta(S_i) \quad \text{for each } S_i \in S \\
x(\ell) &= 0 \quad \text{for each } \ell \in N
\end{aligned}$$

and y^* is the unique solution to the system:

$$\sum_{S_i \in \mathcal{S}} y(S_i)\alpha_\ell(S_i) = 1 \quad \text{for each } \ell \in L \setminus N$$

By arguments similar to those used in the proof of Theorem 3.1, there are two indexes i_1 and i_2, $i_1 \leq i_2$, $i_1, i_2 \in \{0, 1, \cdots, t\}$ such that

- $S_1, \cdots, S_{i_1}$ correspond to the vertices in $G(\mathcal{S}, L \setminus N)$ that have an odd number of edges in the unique path from S_i to the root;
- $S_{i_1+1}, \cdots, S_{i_2}$ correspond to the vertices in $G(\mathcal{S}, L \setminus N)$ that have no path from S_i to the root;
- $S_{i_2+1}, \cdots, S_t$ correspond to the vertices in $G(\mathcal{S}, L \setminus N)$ that have an even number of edges in the unique path from S_i to the root; and
- (S_{i_1}, S_{i_2}) forms a minimum cover.

Since $S_{i_2} \subseteq T^*$, if $\alpha_\ell(T^*) = 1$, then $\alpha_\ell(S_{i_2}) = 1$. Clearly, if $\ell \in N$, then $x^*(\ell) = 0$. If $\ell \notin N$ and $\alpha_\ell(T^*) = 1$ then ℓ must correspond to an edge in a tree component of $G(\mathcal{S}, L \setminus N)$. Therefore, $x^*(\ell) \in \{0, 1\}$ if $\alpha_\ell(T^*) = 1$.

To see (2), observe that by Corollary 3.5, $\alpha_\ell(S^*)x(\ell) = 0$ for each $\ell \in L \setminus L_{T^*}$ and $\alpha_\ell(S^*) = 1$ for each $\ell \in L_{T^*}$. It follows that

$$\alpha(S^*)x = \sum_{\ell \in L_{T^*}} x(\ell) = \beta(S^*).$$

Further, since $x_{L_{T^*}}$ is integral, $cl(x_{L_{T^*}}) = \sigma(supp(x_{L_{T^*}}))$ and so

$$\alpha(cl(x_{L_{T^*}}))x = 2 \sum_{\ell \in L_{T^*}} x(\ell) = 2\beta(S^*) = \beta(cl(x_{L_{T^*}})) \tag{4.6}$$

and

$$\alpha(T^* \vee cl(x_{L_{T^*}}))x = 2\sum_{\ell \in L} x(\ell) = \beta(T^*) + \beta(S^*) = \beta(T^* \vee cl(x_{L_{T^*}})). \tag{4.7}$$

Combining (4.7) and (4.6) we see that $\beta(T^*/cl(x_{L_{T^*}}) = \beta(T^*/S^*)$.

Finally, to see (3), observe that $S^* \subseteq T^* \wedge cl(x_{L_{T^*}})$, but since

$$\beta(T^* \vee cl(x_{L_{T^*}})) + \beta(T^* \wedge cl(x_{L_{T^*}})) \leq \beta(T^*) + \beta(cl(x_{L_{T^*}})),$$

it follows that $\beta(T^* \wedge cl(x_{L_{T^*}})) \leq \beta(S^*)$. □

Corollary 4.10 *Let x be an extreme maximum Matching 2-Lattice vector, (S^*, T^*) be the dominant cover and $e \notin T^*$. If X is a $\beta(S^*)$ intersection in $\mathcal{M}_1(S^*, T^*)$ and $\mathcal{M}_2(S^*, T^*; \{e\})$, then x' defined by*

$$x'(\ell) = \begin{cases} 1 & \text{if } \ell \in X \\ 0 & \text{if } \ell \in L_{T^*} \setminus X \\ x(\ell) & \text{otherwise} \end{cases}$$

is an extreme maximum cardinality Matching 2-Lattice vector with $e \notin cl(x')$.

Proof. In Lemma 4.6, we showed that $x' \in \Omega$. If x' is not extreme, there is a subset $\{x^1, x^2, \cdots, x^k\}$ of distinct vectors in Ω_{ext}, such that

$$x' = \lambda_1 x^1 + \lambda_2 x^2 \cdots + \lambda_k x^k$$

for some $\lambda = (\lambda_1, \lambda_2, \cdots, \lambda_k) > 0$ with $\sum \lambda_i = 1$. We show that $z^i = x_{L_{T^*}} + x^i_{L \setminus L_{T^*}}$ is in Ω for each $i \in \{1, \ldots, k\}$ as follows.

Since $x \in \Omega$, $x_{L_{T^*}}$ satisfies conditions (3) and (4) of Lemma 4.4. Similarly, since $x^i \in \Omega$, $x^i_{L \setminus L_{T^*}}$ satisfies conditions (1) and (2) of Lemma 4.4 for $i = 1, 2, \cdots, k$.

By (2) of Lemma 4.9, $x_{L_{T^*}}$ satisfies (a) of Lemma 4.5. And, since $supp(x^i_{L \setminus L_{T^*}}) \subseteq T^*$,

$$\alpha(cl(x_{L_{T^*}}))x^i_{L \setminus L_{T^*}} = \alpha(cl(x_{L_{T^*}}) \wedge T^*)x^i_{L \setminus L_{T^*}}.$$

But $cl(x_{L_{T^*}}) \wedge T^* = S^*$ so

$$\alpha(cl(x_{L_{T^*}}))x^i_{L \setminus L_{T^*}} = \alpha(S^*)x^i_{L \setminus L_{T^*}} = 0;$$

proving that $x_{L_{T^*}}$ and $x^i_{L \setminus L_{T^*}}$ satisfy condition (b) of Lemma 4.5.

Thus, by Lemma 4.5, z^i is a Matching 2-Lattice vector for each $i \in \{1, \ldots, k\}$. Since

$$x_{L \setminus L_{T^*}} = x'_{L \setminus L_{T^*}} = \lambda_1 x^1_{L \setminus L_{T^*}} + \lambda_2 x^2_{L \setminus L_{T^*}} \cdots + \lambda_k x^k_{L \setminus L_{T^*}}$$

it follows that

$$x = \lambda_1 z^1 + \lambda_2 z^2 \cdots + \lambda_k z^k.$$

Further, since $x'_{L_{T^*}} \in \{0, 1\}$, $x^i_{L_{T^*}} = x'_{L_{T^*}}$ for $i = 1, \ldots, k$. Hence, the members of $\{x^i_{L \setminus L_{T^*}} : i \in [1, \ldots, k]\}$ are distinct and therefore so are the members of $\{z^i : i \in [1, \ldots, k]\}$. This contradicts the assumption that x is extreme. □

Corollary 4.11 *Let (S^*, T^*) be the dominant cover, then*

$$T^* = \cap(cl(x) : x \in \Omega) = \cap(cl(x) : x \in \Omega_{ext}).$$

In the case of matroid intersection, we have the following characterization.

Corollary 4.12 *Let* M_1 *be a matroid with rank function* r_1 *and closure operator* σ_1 *and let* M_2 *be a matroid with rank function* r_2 *and closure operator* σ_2 *both defined on the same ground set* E *and let* Ω_{ext} *be the collection of all maximum cardinality intersections in* M_1 *and* M_2. *Then for each* $I \in \Omega_{ext}$,

$$|I| = r_1(T_1) + r_2(E \setminus T_1) = r_1(E \setminus T_2) + r_2(T_2),$$

where

$$\begin{aligned} T_1 &= \cap(\sigma_1(I) : I \in \Omega_{ext}), \text{ and} \\ T_2 &= \cap(\sigma_2(I) : I \in \Omega_{ext}). \end{aligned}$$

Acknowledgements

Dr. Llewellyn was supported in part by the Office of Naval Research (N00014-89-1658). Dr. Vande Vate was supported in part by the National Science Foundation (DDM-9101581).

References

[1] E. Balas and W. R. Pulleyblank, The Perfectly Matchable Subgraph Polytope of a Bipartite Graph, *Networks* **13** (1983), 495–516.

[2] M. Balinski and K. Spielberg, Methods of Integer Programming: Algebraic, Combinatorial and Enumerative, *in* "Progress in Operations Research," Vol. III, J. Aronofsky, ed. Wiley, New York, 1969.

[3] J. Bartholdi and H. Ratliff, A Field Guide to Identifying Network Flow and Hidden Matching Problems, Research Report No. 77–12 (Department of Industrial and Systems Engineering, University of Florida, Gainesville, 1977).

[4] Wai-Kai Chen, On the Nonsingular Submatrices of the Incidence Matrix of a Graph over the Real Field, *Journal of the Franklin Institute*, **289** (2) (1970), 155–166.

[5] V. Chvátal, Edmonds Polytopes and a Hierarchy of Combinatorial Problems, *Discrete Mathematics* **4** (1973), 305–337.

[6] R. P. Dilworth, A Decomposition Theorem for Partially Ordered Sets, *Annals of Mathematics*, **51** (1950), 161–166.

[7] J. Edmonds, Maximum Matching and a Polyhedron with 0,1 Vertices, *Journal of Research of the National Bureau of Standards (B)* **69** (1965), 125–130.

[8] J. EDMONDS, Submodular Functions, Matroids and Certain Polyhedra, *in* "Combinatorial Structures and their Applications", R. Guy et al., eds., Proceedings of the Calgary International Conference (Gordon and Breach, New York, 1970).

[9] L.R. FORD, JR. AND D.R. FULKERSON, "Flows in Networks", Princeton University Press, Princeton, N.J., 1962.

[10] V. P. GRISHUHIN, Polyhedra related to a Lattice, *Mathematical Programming* **21** (1981), 70–89.

[11] H. GRÖFLIN AND A. HOFFMAN, On Lattice Polyhedra II: Generalization, Construction and Examples, *in* "Algebraic and Geometric Combinatorics", E. Mendelsohn, ed., Annals of Discrete Mathematics **15** North-Holland, Amsterdam, 1982.

[12] H. GRÖFLIN AND T. M. LIEBLING, Connected and Alternating Vectors: Polyhedra and Algorithms, *Mathematical Programming* **20** (1981), 233–344.

[13] A. HOFFMAN, On Lattice Polyhedra III: Blockers and Anti-blockers of Lattice Clutters, *Mathematical Programming Study* **8** (1978), 197–207.

[14] A. HOFFMAN AND H.W. KUHN, Systems of Distinct Representatives and Linear Programming, *The American Mathematical Monthly* **63** (1956), 455–460.

[15] A. HOFFMAN AND D. SCHWARTZ, On Lattice Polyhedra, *in* "Colloquia Mathematica Societatis János Bolyai", **18** Combinatorics, (Keszthely, Hungary, 1976).

[16] E. JOHNSON, On Cut Set Integer Polyhedra, *Cahiers du Centre de Recherche Opérationelle* **17** (1975), 235–251.

[17] D. KÖNIG, Graphs and Matrices, *Matematikai és Fizikai Lapok*, **38** (1931), 116–119.

[18] H.W. KUHN, The Hungarian Method For The Assignment Problem, *Naval Research Logistics Quarterly* **2** (1955), 83–97.

[19] E.L. LAWLER, Matroids With Parity Conditions: A New Class of Combinatorial Optimization Problems, Memorandum Number ERL-M334, *Electronics Research Laboratory* (Berkeley, CA, 1971)

[20] M. MENGER, Zur Allgemeinen Kurventheorie, *Fundamenta Mathematicae* **10** (1927), 96–115.

[21] *G. Nemhauser and L. E. Trotter, Jr.*, Properties of Vertex Packing and Independence System Polyhedra, *Mathematical Programming* **6** (1974), 48–61.

[22] W. R. PULLEYBLANK, "Faces of Matching Polyhedra", Ph.D. Thesis (University of Waterloo, Waterloo, Canada, 1973).

[23] L. S. SHAPLEY AND M. SHUBIK, The Assignment Game I: The Core, *International Journal of Game Theory* **1** (1972), 111–130.

[24] L. E. TROTTER, JR., Solution Characteristics and Algorithms for the Vertex Packing Problem, Technical Report No. 168 (Department of Operations Research, Cornell University, Ithaca, New York, 1973).

[25] J. VANDE VATE, Fractional Matroid Matchings, to appear in *Journal of Combinatorial Theory.*

A PRIMAL DUAL INTEGER PROGRAMMING ALGORITHM

Donna C. Llewellyn
School of Industrial and Systems Engineering, Georgia Institute of Technology, Atlanta, GA
and
Jennifer Ryan
Department of Mathematics, University of Colorado, Denver, CO

submitted to DAM 3/90
Revised 10/90
Accepted (~4/91)

Abstract

We introduce the framework for a primal dual integer programming algorithm. We prove convergence, and discuss some special cases.

1 Introduction

Many optimization algorithms are based on the relationships derived from linear programming duality theory. While Chvátal [5], Blair and Jeroslow [2], Johnson [16] and Wolsey [19] have developed a rich duality theory for integer programming, this theory has not yet been exploited algorithmically. We propose a method which uses Chvátal functions to form a generic primal dual algorithm for general integer programs. When certain subproblems can be solved efficiently, this procedure will solve 0–1 integer programs in time that is pseudopolynomial in the size of the problem. Although, except in special cases, there will not be a polynomial time algorithm to solve the subproblems, they can always be solved by generating cutting planes.

In this section we present some background material. The second section contains a description of the algorithm, and a proof of convergence. In the third section, we discuss some interesting special cases and show how the algorithm generalizes Gomory's familiar cutting plane algorithm. Let Q denote the rational numbers, and let Z denote the integers. Q_+ and Z_+ will denote the nonnegative rationals and integers respectively. Further, $\lfloor x \rfloor$ will denote the greatest integer less than or equal to x. If f is a function, then the function $\lfloor f \rfloor$ is defined by $\lfloor f \rfloor(x) \equiv \lfloor f(x) \rfloor$.

1.1 The Superadditive Dual of an Integer Programming Problem

Let S_n denote the set of n-dimensional superadditive functions. (A function is superadditive if $f(a+b) \geq f(a) + f(b)$ for all a and b.) A duality involving superadditive functions holds for integer programming problems (see e.g. [19]). We briefly outline this duality here.

Consider the integer programming problem

$$\begin{array}{lllll} \text{(P)} & \max & cx & & \\ & \text{s.t.} & Ax & = & b \\ & & x & \geq & 0 \text{ and integer,} \end{array}$$

where A is an integral $m \times n$ matrix, c is an integral n-vector and b is an integral m-vector. If a_j denotes the jth column of A, and c_j is the jth component of c, then the *superadditive dual* of (P) is:

$$\begin{array}{lllll} \text{(D)} & \min & f(b) & & \\ & \text{s.t.} & f(a_j) & \geq & c_j \quad \text{, for j=1,...,n} \\ & & f & \in & S_m. \end{array}$$

The following weak and strong duality properties hold for (P) and (D).

Weak Duality Property: Let x and f be feasible solutions for (P) and (D) respectively. Then $cx \leq f(b)$.

Strong Duality Property: Let x^* be an optimal solution to (P). Then (D) has an optimal solution f^* and $cx^* = f^*(b)$.

The Weak Duality Property is easily verified using superadditivity. The Strong Duality property follows from the fact that the value function of (P) is superadditive.

Blair and Jeroslow ([2]) have shown that the duality still holds even when the class of functions is restricted to be *Chvátal Functions*. The class of Chvátal functions can be defined as follows. Let L_n denote the set of n-dimensional linear functions with rational coefficients.

Definition: The class C_n of n-dimensional Chvátal functions is the smallest class K satisfying the following properties:

1. If $f \in L_n$ then $f \in K$;
2. If $f, g \in K$ and $\alpha, \beta \in Q_+$, then $\alpha f + \beta g \in K$;
3. If $f \in K$, then $\lfloor f \rfloor \in K$.

Note that a Chvátal function is superadditive. and that the class of Chvátal functions contains the linear functions. The Weak Duality Property still holds since the Chvátal functions are superadditive. That the Strong Duality Property still holds for this restricted class is nontrivial, and the interested reader is referred to [2].

1.2 A Separation Theorem and Optimality Conditions

As a consequence of strong duality the following proposition holds. (See [2], or derive Proposition 1.2.1 by applying the Strong Duality Property to an integer programming problem with artificial variables and a phase 1 objective.)

Proposition 1.2.1 [2] *Let A be an $m \times n$ matrix with integer entries, and let b be an integral m-vector. Then exactly one of the following alternatives holds:*

1. *There exists $x \in Z_+^n$ with $Ax = b$;*
2. *There exists $f \in C_m$ with $f(a_j) \geq 0$ for all $j = 1, \ldots, n$ and $f(b) < 0$.*

Blair and Jeroslow ([3]) show that the separating function of alternative 2 may have an exponential nesting of round-downs, so that Proposition 1 does not give an **NP∩co-NP** characterization of the integer programming feasiblity problem.

We conclude this section by presenting the optimality conditions for (P) and (D) (given here as Proposition 1.2.2). Similar conditions involving general superadditive functions are given in [16] and [19]. We omit the straightforward proof of Proposition 1.2.2.

Proposition 1.2.2 *Let x and f be feasible solutions to the problems* (P) *and* (D) *given above. Then x and f are optimal solutions if and only if the following two conditions hold:*

1. **(Complementary Slackness)** *For all $j = 1, \ldots, n$, if $f(a_j) > c_j$ then $x_j = 0$;*
2. **(Complementary Linearity)** $\sum_{j=1}^{n} f(a_j)x_j = f(b)$.

Note that Condition 1 is analogous to the usual complementary slackness conditions of linear programming. In the event that f is a linear function, Condition 2 holds trivially.

2 The Algorithm

In this section we introduce our primal dual algorithm using the duals introduced in the last section. There will be a couple of places where the actual details will be left as "black boxes." This is done for several reasons. Primarily, this allows us to present the algorithm as a generic framework into which many different implementations may be built. We will prove what is necessary for these black boxes to do in order to assure finite convergence of the algorithm and then in the next section we discuss a few implementations that satisfy these requirements.

2.1 The Basic Steps

As mentioned in the last section, we always will assume that all data is integral.

PRIMAL DUAL ALGORITHM

Input: Integral $m \times n$ matrix A, integral n-vector c and integral nonnegative m-vector b.

Ouput: Integral nonnegative n-vector x optimizing integer programming problem (P), or information that (P) is infeasible or unbounded; and a Chvátal function f optimizing the dual problem (D), or information that (D) is infeasible or unbounded.

Step 1: Let f be a dual feasible function. Without loss of generality, assume that $f = \lfloor f \rfloor$. Let $J = \{j \in \{1, \ldots, n\} | f(a_j) = c_j\}$. [If no such f exists then (D) is infeasible, and (P) is infeasible or unbounded. STOP.]

Step 2: Consider the integer program that looks for an integral solution to (P) using only coordinates in the set J. Call this problem (RFP) [restricted feasibility problem]:

$$\begin{aligned} \text{(RFP)} \quad \max \quad W &= \sum_{i=1}^{m} -x_i^a \\ \text{s.t.} \quad & \sum_{j \in J} a_{ij} x_j + x_i^a = b_i \text{ for } i = 1, \ldots, m \\ & \text{all } x \geq 0 \text{ and integer.} \end{aligned}$$

There are three possibilities:

- $W^* < 0$. Go to Step 3.
- $W^* = 0$ and $\sum_{j \in J} f(a_j) x_j^* = f(b)$. Go to Step 6.
- $W^* = 0$ and $\sum_{j \in J} f(a_j) x_j^* < f(b)$. Go to Step 7.

Step 3: From Proposition 1.2.1 it is clear that there exists a Chvátal function $\bar{f}$ such that:

$$\begin{aligned} \bar{f}(a_j) &\geq 0 \quad \forall j \in J, \\ \bar{f}(b) &\leq -1. \end{aligned}$$

There are two possibilities:

- $\exists$ some $j \notin J$ with $\bar{f}(a_j) < 0$. Go to Step 4.
- $\bar{f}(a_j) \geq 0 \; \forall j$. Go to Step 5.

Step 4:

$$\text{Let } \theta = \min_{\substack{j \notin J \\ \bar{f}(a_j)<0}} \left\{ \frac{c_j - f(a_j)}{\bar{f}(a_j)} \right\}.$$

Then set $\hat{f} = \left\lfloor f + \theta \bar{f} \right\rfloor$. Replace f by $\hat{f}$ and update J. Go to Step 2.

Step 5: Here it is clear that (D) is unbounded and hence (P) is infeasible. STOP.

Step 6: Here x^* and f satisfy the optimality conditions and so are optimal. STOP.

Step 7: Now consider the following problem that attempts to push the objective up to $f(b)$ while maintaining integrality. This problem is called (RMP) [restricted maximization problem]:

$$\begin{aligned} \text{(RMP)} \quad \max \quad V \;&=\; \sum_{j \in J} f(a_j) x_j \\ \text{s.t.} \quad & \sum_{j \in J} a_{ij} x_j = b_i \text{ for } i = 1, \ldots, m \\ & \sum_{j \in J} f(a_j) x_j \leq f(b) \\ & \text{all } x \geq 0 \text{ and integer} \end{aligned}$$

There are two possibilities:

- $V^* = f(b)$ and all x_j^* for $j \in J$ are integral, where x^* is the optimal solution of (RMP). Go to Step 6.
- Otherwise (there is no integer solution with value $f(b)$) go to Step 8.

Step 8: It is known by the strong duality property that there exists a Chvátal function $\bar{f}$ such that

$$\begin{aligned} \bar{f}(a_j) &\geq f(a_j) \quad \forall j \in J, \\ \bar{f}(b) &< f(b). \end{aligned}$$

There are two possibilities:

- If $\bar{f}(a_j) \geq c_j \ \forall j$ then $\bar{f}$ is dual feasible. Replace f by $\lfloor \bar{f} \rfloor$ and update J. Go to Step 2.
- Otherwise, go to Step 9.

Step 9:

$$\text{Let } \lambda = \max_{\substack{j \notin J \\ c_j > \bar{f}(a_j)}} \left\{ \frac{c_j - \bar{f}(a_j)}{f(a_j) - \bar{f}(a_j)} \right\}$$

and set $\hat{f} = \lfloor \lambda f + (1-\lambda)\bar{f} \rfloor$. Then, $\hat{f}$ is a dual feasible function. Replace f by $\hat{f}$ and update J. Go to Step 2.

We conclude this subsection by noting that the algorithm only requires storing the values of $f(a_j)$, for $1 \leq j \leq n$, and $f(b)$, which can be updated as the algorithm progresses. It is not necessary to store a representation of the entire function f.

2.2 Correctness and Finite Convergence

Now let us look in detail at some of the steps of the algorithm defined in Section 2.1. In Step 1, it is not difficult to get an initial dual feasible function; in particular, one can solve the linear programming relaxation of (P) and use the (linear) dual function. Since we have assumed that all data is integral, it is clear that if the linear programming dual is infeasible then (D) is also infeasible. Denote by f_0 the function used by Step 1. In Step 2, it is clear that (RFP) is always feasible (set $x_i^a = b_i$ for all i and $x_j = 0$ for all $j \in J$). Further, as all the x_i^a are nonnegative, it is clear that the objective will always be nonpositive.

Consider θ arising in Step 4. By construction it is clear that the denominator is less than zero. Further, by dual feasibility of f and the definition of J, the numerator is also negative, hence θ will be positive. Now, since a positive linear combination of Chvátal functions is a Chvátal function, $\hat{f}$ will be a Chvátal function. Hence, by the construction in Step 3, $\hat{f}$ will be dual feasible so it is justified to go to Step 2 with $\hat{f}$. Note that by Step 3, $\bar{f}(b) < 0$ and hence $\hat{f}(b) < f(b)$. Further, by definition, $\hat{f}$ is integral so $\hat{f}(b) \leq f(b) - 1$.

The algorithm reaches Step 5 when the $\bar{f}$ found in Step 3 satisfies $\bar{f}(a_j) \geq 0 \ \forall j$. Consider the function $\tilde{f} = \lfloor f + \epsilon \bar{f} \rfloor$. Clearly, this function will be dual feasible for any nonnegative ϵ. Further, as ϵ approaches infinity, $\tilde{f}(b)$ approaches negative infinity. Hence the statement

made in Step 5 is correct, i.e., (D) is unbounded. By Weak Duality it is clear then that (P) is infeasible.

Step 6 can reached in two ways. In either case (from Step 2 or Step 7), we have an x^* that satisfies (P) for which $x_j^* > 0 \Rightarrow f(a_j) = c_j$, so complementary slackness holds (Condition 1 of Proposition 1.2.2). Further, we know that $\sum_{j \in J} c_j x_j^* = \sum_{j \in J} f(a_j) x_j^* = f(b)$. Hence complementary linearity holds (Condition 2 of Proposition 1.2.2). Thus by Proposition 1.2.2, x^* and f are optimal.

In Step 8, clearly if $\bar{f}(a_j) \geq c_j$ for all j then $\bar{f}$ is dual feasible so it is acceptable to return to Step 2. Further, note that by construction $\bar{f}(b) < f(b)$ and since $\lfloor \bar{f} \rfloor$ is integral, $\lfloor \bar{f}(b) \rfloor \leq f(b) - 1$.

Consider λ found in Step 9. Clearly by construction the numerator is positive. Further, since f is dual feasible we know that $f(a_j) \geq c_j \ \forall j$ and since we also know by construction that within the definition of λ, $\bar{f}(a_j) < c_j$, it is clear that $0 < \lambda < 1$. Thus, since a convex combination of Chvátal functions is a Chvátal function, $\hat{f}$ is a Chvátal function. It is not difficult to see that $\hat{f}$ is dual feasible and hence it is acceptable to return to Step 2 with $\hat{f}$. Further, since both $\bar{f}(b)$ and (hence) $\hat{f}(b)$ are less than $f(b)$, and since $\hat{f}$ is integral, it is clear that $\hat{f}(b) \leq f(b) - 1$.

Now consider the flow of the algorithm. Notice that each time the algorithm leaves Step 2, it either ends up in Step 4, 5, 6, 8 or 9. Steps 5 and 6 are terminal steps as shown above. Further we have shown that when the algorithm leaves Steps 4, 8 or 9 to return to Step 2, it does so with an integral dual feasible function which forces the objective value to decrease by at least 1. Hence, if f^* is the optimal dual function, then in no more than $f_0(b) - f^*(b)$ steps the algorithm must terminate. If f_0 corresponds to the dual solution of the linear programming relaxation of (P), then $f_0(b) = \lfloor cx^* \rfloor$, where x^* is the optimal linear programming solution. If (P) is a 0-1 integer programming problem, $f_0(b) - f^*(b) \leq \sum_{j=1}^{n} c_j$. Thus if the subproblems can be solved in polynomial (or pseudopolynomial) time, a pseudopolynomial time algorithm results.

We should note here that in Step 7, it is not necessary to solve (RMP) to optimality. It is merely necessary to find a dual solution of value less than $f(b)$ (which proves that the value of (RMP) is less than $f(b)$). (Note that the dual of any linear programming solution of (RMP) will have value at *most* $f(b)$.) This fact results in computational savings when Step 7 is implemented using cutting planes, as discussed in Section 3.2.

3 The Black Boxes

The major problems with implementing the primal dual algorithm of Section 2 above are in Steps 2, 3, 7 and 8. In particular, note that (RFP) is an integer programming problem that could be as difficult as the original problem (P). Here, we discuss two approaches to dealing with solving (RFP) and implementing the black boxes of Steps 3 and 8 of the algorithm.

In Section 3.1, we look at two special cases of (P) where the structure allows for an easy solution of (RFP). Then, in Section 3.2, we consider cutting planes. Specifically we specialize the results of Chvátal ([5]) to build dual superadditive functions using Gomory cutting planes in order to implement Steps 2, 3, 7, and 8 of the algorithm. Cutting planes can always be used to solve (RFP) and (RMP) if no special purpose methods are available.

Finally, in Section 3.3 we show how our framework can be used to find a maximum weight matching in a graph. In this case the cardinality matching algorithm of Edmonds ([7]) is used to solve (RFP).

3.1 Easy Special Cases

Here we briefly discuss two very special cases where implementation of the algorithm is easy. For any $J \subseteq \{1, \ldots, n\}$ define (PJ) to be the subproblem of (P) using only the columns of A in the set J.

First consider the case when (PJ) is feasible if and only if its linear programming relaxation is feasible for each J. In this case, $\{x \in Q^n | Ax = b, x \geq 0\}$ is an integral polyhedron, and we will show that our algorithm reduces to the usual primal dual algorithm for linear programming. Since (PJ) is feasible if and only if its linear programming relaxation is feasible, $W^* < 0$ if and only if the value of the linear programming relaxation of (RFP) is less than 0. If so, then there is a linear function $\bar{f}$ (the optimal dual solution to the linear programming relaxation of (RFP)) that will satisfy the conditions of Step 3 of the algorithm. In general, the rounding of the function that occurs in Step 4 of the algorithm is vital to the convergence of the algorithm. However in this case, because the linear dual solution can be used, convergence is guaranteed without rounding. As with the usual primal dual algorithm for *linear* programming, we can assume all optimal solutions to (RFP) are basic solutions. Since no basis is ever repeated, the algorithm is finite. Thus, with the rounding step omitted, the dual function will remain linear throughout the execution of the algorithm. As a

consequence, complementary linearity will always be satisfied trivially, (RMP) need never be solved, and our algorithm reduces to the primal dual algorithm for linear programming.

Next, define the group relaxation of (P), called here (G):

$$\begin{array}{llll} \text{(G)} & \max & cx & \\ & \text{s.t.} & Ax & = b \\ & & x & \text{integer.} \end{array}$$

As our second easy special setting, we now consider the case when for any column set, J, the problem (PJ) is feasible if and only if both its linear relaxation and group relaxation are feasible. (A simple example where this condition holds is if A is a diagonal matrix with nonnegative integers along its diagonal.) Now, to solve (RFP) one must first check if the linear relaxation is feasible. If not, then as above, the linear programming dual solution will work in Step 3 of the algorithm directly. If the linear relaxation is feasible, then next check the feasibility of the group relaxation. If the group relaxation is not feasible, by the theorem of the alternative for integral systems of equations (see [8] for example), it is known that

$$\begin{array}{lllll} \text{There exists } y \text{ such that} & ya_j & \in & Z & \forall j \in J, \\ \text{but} & yb & \notin & Z. & \end{array}$$

Moreover, y can be found in polynomial time by a unimodular elimination scheme (see [6]). Then, let $\bar{f}(w) = \lfloor yw \rfloor - yw$. This function satisfies the property required by Step 3 of the algorithm. If both the linear and group relaxations are feasible, then there is a feasible solution to (RFP) with value 0, and this solution will be optimal for (P). In particular, there will be an optimal solution to the linear programming relaxation of (RFP) that is integral. This integral solution can be found by pivoting among the alternate optima to the linear programming relaxation or by using other standard techniques. Note that such an integral solution need only be sought in the final iteration of the algorithm.

For any integer programming problem, the first instance of (RFP) to arise can always be easily solved in the absence of dual degeneracy in the linear programming relaxation. Usually the linear programming relaxation of (P) is solved in Step 1 of the algorithm to find f_0; and then (RFP) is solved. In the absence of dual degeneracy the set J is exactly equal to the indices of the basic variables of the optimal solution of the linear programming relaxation. Hence, unless the solution of the linear programming problem is integral, (and thus the optimal integer solution), (RFP) is infeasible. If B is the basis, there is some row of B^{-1}, say β, such that βb is not integer. Since for every $j \in J$, a_j is a column of B, βa_j is

either 0 or 1. Thus the function $\bar{f}(w) = \lfloor \beta w \rfloor - \beta w$ satisfies the property required by Step 3 of the algorithm.

3.2 Cutting Planes

Both (RFP) and the integer programming restriction of (RMP) can always be solved using cutting planes. In [5] Chvátal has shown the following (although using different terminology):

Theorem 3.2.1 (Chvátal) *Consider the integer programming problem* $\max\{cx | Ax \leq b,\ x \text{ integer}\}$. *If* $\sum_{j=1}^{n} \alpha_j^k x_j \leq \beta^k$, $k = 1, \ldots, r$, *is a series of cuts constructed by Gomory's cutting plane algorithm, then there are Chvátal functions* F^k, $k = 1, \ldots, r$, *such that* $F^k(a_j) = \alpha_j^k$ *and* $F^k(b) = \beta^k$, *for all* $k = 1, \ldots, r$.

Theorem 3.2.1 can easily be adapted to our setting.

Theorem 3.2.2 *Consider the restricted feasibility problem* (RFP). *If* $\sum_{j \in J} \alpha_j^k x_j + \sum_{i=1}^{m} \gamma_i^k x_i^a \leq \beta^k$, $k = 1, \ldots, r$, *is a series of cuts constructed by Gomory's cutting plane algorithm, then there are Chvátal functions* F^k, $k = 1, \ldots, r$, *such that* $F^k(a_j) = \alpha_j^k$ *for all* $j \in J$, $F^k(e_i) = \gamma_i^k$ *and* $F^k(b) = \beta^k$, *for all* $k = 1, \ldots, r$.

Similarly, consider the restricted maximization problem (RMP). *If* $\sum_{j \in J} \alpha_j^k x_j \leq \beta^k$, $k = 1, \ldots, r$, *is a series of cuts constructed by Gomory's cutting plane algorithm, then there are Chvátal functions* F^k, $k = 1, \ldots, r$, *such that* $F^k(a_j) = \alpha_j^k$ *for all* $j \in J$ *and* $F^k(b) = \beta^k$, *for all* $k = 1, \ldots, r$.

For continuity of exposition, the details of the construction of the function guaranteed by Theorem 3.2.2 are left until the end of this subsection. We now show how such functions can be used to solve both (RFP) and (RMP). (See also [19] for results relating cutting planes and Chvátal functions.)

Recall (RFP), written as an optimization problem with artificial variables.

$$\begin{aligned} \max \quad W \quad &= \quad \sum_{i=1}^{m} -x_i^a \\ \text{s.t.} \quad &\quad \sum_{j \in J} a_{ij} x_j + x_i^a = b_i, \text{ for } i = 1, \ldots, m, \\ &\quad \text{all } x \geq 0 \text{ and integer.} \end{aligned}$$

If the linear programming relaxation of (RFP) has an optimal solution with value less than 0, let y be the optimal dual solution. Then from linear programming duality, $ya_j \geq 0$ for all $j \in J$ and $yb < 0$. Define $\bar{f}(w) \equiv yw$.

Otherwise, the optimal value of the linear programming relaxation of (RFP) is 0. In this case we generate cuts until either

1. An integer solution is obtained, and we return to Step 2 of the algorithm with optimal solution x^*, and $W^* = 0$; or

2. We finally get a solution to the linear programming problem with added cuts, whose value is negative. That is, the following linear programming problem has value less than 0:

$$\begin{array}{lll} \max & \sum_{i=1}^{m} -x_i^a & \\ \text{s.t.} & \sum_{j \in J} a_{ij}x_j + x_i^a = b_i, & \text{for } i = 1, \ldots, m \\ & \sum_{j \in J} F^k(a_j)x_j + \sum_{i=1}^{m} F^k(e_i)x_i^a \leq F^k(b), & \text{for } k = 1, \ldots, r \\ & \text{all } x \geq 0 & \end{array}$$

where the functions F^k, for $k = 1, \ldots, r$, are the Chvátal functions corresponding to the added cuts as in Theorem 3.2.2. Let (y, μ) be the optimal dual solution to this linear programming problem, where y is the vector of dual variables corresponding to the original constraints, and μ is the vector of dual variables corresponding to the added cutting planes. Note that since the cutting-plane constraints are inequalities, we must have $\mu \geq 0$. By linear programming duality, $ya_j + \sum_{k=1}^{r} \mu_k F^k(a_j) \geq 0$ for all $j \in J$ and $yb + \sum_{k=1}^{r} \mu_k F^k(b) < 0$. Thus we can let $\bar{f}(w) \equiv yw + \sum_{k=1}^{r} \mu_k F^k(w)$. Since $\mu \geq 0$, $\bar{f}$ is in C_m and it satisfies the required conditions of Step 3 of the algorithm.

The integer programming restriction of Problem (RMP) can be handled similarly, with the understanding that the constraint $\sum_{j \in J} f(a_j)x_j \leq f(b)$ of (RMP) should be treated just as any other added cut. Also recall that it is not necessary to find an integer optimal solution of (RMP), only to find an $\bar{f}$ with $\bar{f}(a_j) \geq f(a_j)$ for all $j \in J$, and $\bar{f}(b) < f(b)$. If the optimal solution to (RMP) is less than $f(b)$, then let y be the optimal dual solution. So, $ya_j \geq f(a_j)$ for all $j \in J$ and $yb < f(b)$. In this case we can let $\bar{f}(w) \equiv yw$. Otherwise, add cuts with corresponding Chvátal functions $F^1, \ldots, F^r$, until either

1. An integer optimum is obtained with value $f(b)$. In this case the solution is optimal, and we return from (RMP) with x^*; or

2. An optimal linear programming solution is reached (integer or fractional) with value less than $f(b)$. As with (RFP), we let (y, μ) be the optimal dual solution and let $\bar{f}(w) \equiv yw + \sum_{k=1}^{r} \mu_k F^k(w)$. By linear programming duality we have that $\mu \geq 0$, so $\bar{f}$ is a Chvátal function, and that $\bar{f}(a_j) \geq f(a_j)$ for all $j \in J$ and $\bar{f}(b) < f(b)$ so that the conditions of Step 8 of the algorithm are satisfied.

We now give the details of the construction of the function F corresponding to a cut as in Theorem 3.2.2. Although these can be derived from [5] we have included them for completeness. We will asume that we are working on (RFP). The adaptation for (RMP) is straightforward: the same procedure is followed with the omission of the artificial variables.

Suppose that we have already added r cuts, so that the system we currently have is

$$\begin{aligned} \sum_{j \in J} a_{ij} x_j + x_i^a &= b_i, \quad \text{for } i = 1, \ldots, m \\ \sum_{j \in J} F^k(a_j) x_j + \sum_{i=1}^{m} F^k(e_i) x_i^a &\leq F^k(b), \quad \text{for } k = 1, \ldots, r \\ \text{all } x \geq 0 &\text{ and integer.} \end{aligned}$$

Let the slack variables which have been added to the tableau for the rows corresponding to the cuts be $s_1, \cdots, s_r$. We can suppose without loss of generality that the F's being constructed are always rounded down, and thus always have integer values. Suppose that we want to cut on row i of the current tableau, and that row i has the form:

$$\sum_{j \in J} \eta_j x_j + \alpha_1 x_1^a + \cdots + \alpha_m x_m^a + \gamma_1 s_1 + \cdots + \gamma_r s_r = \delta \tag{1}$$

where δ is fractional. The cut we wish to generate is

$$\sum_{j \in J} \lfloor \eta_j \rfloor x_j + \lfloor \alpha_1 \rfloor x_1^a + \cdots + \lfloor \alpha_m \rfloor x_m^a + \lfloor \gamma_1 \rfloor s_1 + \cdots + \lfloor \gamma_r \rfloor s_r \leq \lfloor \delta \rfloor. \tag{2}$$

Note that in the usual statement of Gomory's algorithm, the cut used is (2) - (1), instead of (2). However adding (2) alone has the same effect, since (1) is an equality and remains part of the problem. Solving for the slack s_l in (1), and substituting in (2) gives

$$\sum_{j \in J} \left(\lfloor \eta_j \rfloor - \sum_{k=1}^{r} \lfloor \gamma_k \rfloor F^k(a_j) \right) x_j + \sum_{i=1}^{m} \left(\lfloor \alpha_i \rfloor - \sum_{k=1}^{r} \lfloor \gamma_k \rfloor F^k(e_i) \right) x_i^a$$

$$\leq \lfloor \delta \rfloor - \sum_{k=1}^{r} \lfloor \gamma_k \rfloor F^k(b). \tag{3}$$

Let the current basis be B, and let the ith row of B^{-1} be $B_i^{-1} = (\beta_1, \cdots, \beta_m, \gamma_1, \cdots, \gamma_r)$. Note that the coefficients of the slacks in (1) will always occur in the ith row of B^{-1}, since there were originally unit vectors in these columns. We thus have that

$$\eta_j = B_i^{-1} \begin{pmatrix} a_{1j} \\ a_{2j} \\ \cdot \\ \cdot \\ a_{mj} \\ F^1(a_j) \\ \cdot \\ \cdot \\ F^r(a_j) \end{pmatrix}$$

$$= (\beta_1, \ldots, \beta_m) a_j + \sum_{k=1}^{r} \gamma_k F^k(a_j), \; j = 1, \ldots, m. \tag{4}$$

Similarly

$$\alpha_i = (\beta_1, \cdots, \beta_m) e_i + \sum_{k=1}^{r} \gamma_k F^k(e_i), \; i = 1, \ldots, m, \tag{5}$$

and

$$\delta = (\beta_1, \cdots, \beta_m) b + \sum_{k=1}^{r} \gamma_k F^k(b). \tag{6}$$

We now define,

$$F(w) \equiv \left\lfloor (\beta_1, \cdots, \beta_m) w + \textstyle\sum_{k=1}^{r} (\gamma_k - \lfloor \gamma_k \rfloor) F^k(w) \right\rfloor. \tag{7}$$

Since $\gamma_k - \lfloor \gamma_k \rfloor \geq 0$ always, F is a Chvátal function. Also, since $F^k(w)$ is always forced to be integer, we can take $\sum_{k=1}^{r} \lfloor \gamma_k \rfloor F^k(w)$ outside the round-down sign and write:

$$F(w) = \left\lfloor (\beta_1, \cdots, \beta_m) w + \textstyle\sum_{k=1}^{r} \gamma_k F^k(w) \right\rfloor - \sum_{k=1}^{r} \lfloor \gamma_k \rfloor F^k(w). \tag{8}$$

Using (4), (5), (6) and (8), it is easy to verify that (7) generates the cut (3).

Notes: If cutting planes are being used to solve the "black boxes," cuts generated in (RFP) should not be discarded (except for terms corresponding to the artificial variables) when proceeding to (RMP). They will remain valid.

Moreover, when solving (RMP), it is not necessary to continue to add cuts until the optimal integer solution is found. It suffices to add cuts only until the optimal fractional

value falls below $f(b)$. Then the dual function $\bar{f}$ can be constructed as above, and it will have value less than $f(b)$ as required by Step 8 of the algorithm.

It is important to note that if one is solving a problem for which some or all of the facet defining inequalities (or some other "deep" cuts) are known, then these can be used in place of the Gomory cuts since it has been proven (see [4], [13] and [18]) that every valid inequality of (P) is equivalent to or dominated by a an inequality generated by a Chvátal function. Of course, one must still construct this function from the facet defining inequality.

3.3 Matching

The primal dual algorithm of Section 2 can be specialized to find a maximum weight matching in a weighted graph G. We have chosen this example because (RFP) can be solved efficiently. When it becomes necessary to solve (RMP), deep cuts can be generated using a separation procedure due to Padberg and Rao ([17]). This approach is not likely to be more efficient than specialized weighted matching algorithms, but provides a nice illustration of how the primal dual algorithm can be tailored to a specific problem.

Without loss of generality suppose that any maximum weight matching is a perfect matching (G is easily altered so that this is the case). Then, if A is the node-edge incidence matrix of G, b is the vector of all 1's, and c is the vector of edge weights, the problem of finding a maximum weight matching is exactly the problem (P) of Section 1. Let the node set of G be V.

Definition: Given a graph G, an *odd set cover* of G is a set of node sets $N_1, \ldots, N_s$ each having odd cardinality greater than 1, and a set of singletons $v_1, \ldots, v_r$ such that every edge of G either has both endpoints contained in some N_k or is incident to some v_j. The capacity of the cover is equal to $r + \sum_{k=1}^{s} \lfloor \frac{|N_k|}{2} \rfloor$.

Edmonds (see [7]) showed that the following duality holds: the maximum cardinality of a matching in a graph is equal to the minimum capacity of an odd set cover. The algorithm given in [7] returns both a maximum cardinality matching and the corresponding odd set cover. We will use this matching duality result here, and assume that we have available an algorithm that finds a maximum cardinality matching which also returns an odd set cover whose capacity is equal to the cardinality of the matching.

Problem (RFP) for the matching problem is efficiently solved. Given J, we must determine if (P) is feasible using only the variables in J. In the matching setting, we must

determine if there is a perfect matching in G using only the edges indexed by J. We begin by finding the maximum cardinality matching on the graph whose edges consist only of those indexed by J. If this is a perfect matching of G, then we have a feasible solution of (P) satisfying complementary slackness and we proceed to Step 6 or 7 of the algorithm. Otherwise, we have a matching of size p, where $2p$ is less than the number of nodes in G. The cardinality matching algorithm will also have returned an odd set cover of the edges indexed by J with capacity p. Let the odd set cover be $\{N_1, N_2, \ldots, N_s, v_1, \ldots, v_r\}$. Without loss of generality assume that the vertices $v_1, \ldots, v_r$ correspond to the first r rows of A.

Theorem 3.3.1 *The function*

$$f(w) = \sum_{j=1}^{r} w_j + \sum_{k=1}^{s} \lfloor \sum_{i \in N_k} \frac{w_i}{2} \rfloor - \frac{1}{2} \sum_{i \in V} w_i$$

satisfies the condition of Step 3 in the algorithm.

Proof: We must show that $f(a_j) \geq 0$ for each $j \in J$ and that $f(b) < 0$. Recalling that b is the vector of all 1's, the negative term in $f(b)$ is $-\frac{|V|}{2}$. The positive term will be equal to $r + \sum_{k=1}^{s} \lfloor \frac{|N_k|}{2} \rfloor$, the capacity of the cover returned by the cardinality matching algorithm. Since the capacity of the cover is equal to the cardinality of the matching found, and the matching was not perfect, $f(b) < 0$.

Now let e be an edge indexed by $j \in J$. Then a_j has a 1 in the positions corresponding to the two endpoints of e and 0's elsewhere. Since e is indexed by a member of J, it is covered by the odd set cover given by the cardinality matching algorithm. If e is incident with one of the v_i's then $f(a_j) \geq 1 - 1 = 0$. If e has both its endpoints contained in N_k, then $\lfloor \sum_{i \in N_k} \frac{(a_j)_i}{2} \rfloor = 1$ and $f(a_j) \geq 1 - 1 = 0$. □

Thus (RFP) can be completely solved through the use of the cardinality matching algorithm. We now consider (RMP). When we reach step 7 of the algorithm we have found a perfect matching using only edges in J, but the value of that perfect matching is less than $f(b)$.

When solving (RMP), it is necessary to find an integer solution of value $f(b)$ or a dual solution f' with $f'(b) < f(b)$. As discussed above, this can be accomplished by adding cutting planes until the value of (RMP) first drops below $f(b)$. (Recall that (RMP) will always have value at most $f(b)$.) In the matching setting we have the advantage that we know what the facets of the corresponding polytope are.

Let S be a set of nodes in the graph, and let $E(S)$ denote the set of edges whose both endpoints are in S. Then every facet-defining inequality is of the form

$$\sum_{j \in E(S)} x_j \leq \frac{|S| - 1}{2},$$

where S is a node set of odd cardinality. It easy to see that the function

$$f(w) \equiv \lfloor \sum_{i \in S} \frac{w_i}{2} \rfloor$$

defines the facet derived from an odd set S.

Now suppose that we have solved the linear programming relaxation of (RMP), and obtained a basic fractional solution with value equal to $f(b)$. The fractional solution will violate one of the above facet describing inequalities. Padberg and Rao [17] showed that the separation problem for the matching polytope can be solved in polynomial time. That is, a facet-defining inequality that is violated by a given infeasible solution can be determined in polynomial time. Thus in polynomial time we can generate a cut as described above that will reduce the value of (RMP) as desired, or find an integer solution with value equal to $f(b)$.

4 Future Work

Future work related to this algorithm should be directed towards finding special structures which allow efficient solution of the restricted subproblems. More specifically, one aim is to identify cases where the problem can be solved without resorting to the use of cutting planes in the solution of (RFP). These problems can be of two types – those where the type of data is known to be such that the subproblems can be solved easily (like those cases discussed in Section 3.1 for example), and those where the problem structure provides other avenues for solution (as in Section 3.3). With respect to this last type of problem, the richest potential lies with integer programming problems with known pseudopolynomial algorithms. It would, of course, be most rewarding to use our framework to establish pseudopolynomial algorithms for classes of integer programs for which there do not yet exist "efficient" solution procedures. One other direction for future work is in the area of column generation. In problems with exponentially many columns, it is expected that our procedure will not only keep the number of active columns small but also will direct the user to a "smart" set of such columns.

References

[1] C.E. Blair and R.G. Jeroslow (1977). The Value Function of a Mixed Integer Program 1, *Discrete Mathematics* **24**, 121-138.

[2] C.E. Blair and R.G. Jeroslow (1982). The Value Function of an Integer Program, *Mathematical Programming* **23**, 237-273.

[3] C.E. Blair and R.G. Jeroslow (1986), Computational Complexity of Some Problems in Parametric Discrete Programming. I, *Mathematics of Operations Research* **11.2**, 241-260.

[4] C.A. Burdet and E.L. Johnson (1977). A Subadditive Approach to Solve Linear Integer Programs, *Annals of Discrete Mathematics* **1**, 117-144.

[5] V. Chvátal(1973). Edmonds Polytopes and a Hierarchy of Combinatorial Problems, *Discrete Mathematics* **4**, 305-337.

[6] P.D. Domich, R. Kannan, and L.E. Trotter, Jr. (1987), Hermite Normal Form Computation Using Modulo Determinant Arithmetic, *Mathematics of Operations Research* **12**, 50-59.

[7] J. Edmonds (1965). Paths, Trees and Flowers, *Canadian J. Math.* **17**, 449-467.

[8] J. Edmonds and F.R. Giles (1977). A Min-Max Relation for Submodular Functions on Graphs, *Annals of Discrete Mathematics* **1**, 185-204.

[9] R.E. Gomory (1958). Outline of an Algorithm for Integer Solutions to Linear Programs, *Bulletin of the American Mathematical Society* **64**, 275-278.

[10] R.E. Gomory (1960). Solving Linear Programs in Integers, in *Combinatorial Analysis*, R.E. Bellman and M. Hall, Jr., eds., American Mathematical Society, pp. 211-216.

[11] R.E. Gomory (1963). An Algorithm for Integer Solutions to Linear Programs, in *Recent Advances in Mathematical Programming*, R. Graves and P. Wolfe, eds., McGraw-Hill, pp. 269-302.

[12] R.E. Gomory (1963). An All-Integer Programming Algorithm, in *Industrial Scheduling*, J.F. Muth and G.I. Thompson, eds., Prentice-Hall, pp. 193-206.

[13] R.G. Jeroslow (1978). Cutting Plane Theory: Algebraic Methods, *Discrete Mathematics* **23**, 121-150.

[14] R.G. Jeroslow (1979). An Introduction to Cutting Planes, *Annals of Discrete Mathematics* **5**, 71-95.

[15] E.L. Johnson (1973). Cyclic Groups, Cutting Planes and Shortest Paths, in *Mathematical Programming*, T.C. Hu and S. Robinson, eds., Academic Press, pp. 185-211.

[16] E.L. Johnson (1979). On the Group Problem and a Subadditive Approach to Integer Progamming, *Annals of Discrete Mathematics* **5**, 97-112.

[17] M.W. Padberg and M.R. Rao (1982). Odd Minimum Cut-Sets and b-Matchings, *Mathematics of Operations Research* **7**, 67-80.

[18] A. Schrijver (1980). On Cutting Planes, *Annals of Discrete Mathematics* **9**, 291-296.

[19] L.A. Wolsey (1981). Integer Programming Duality: Price Functions and Sensitivity Analysis, *Mathematical Programming* **20**, 173-195.

[20] L.A. Wolsey (1981). The b-Hull of an Integer Program, *Discrete Applied Mathematics* **3** 193-201.